Grado en Ingeniería Informática

Trabajo Final de Grado

Proyecto-NFC-RRSS

Autor:
Alejo Molés Ramos

Supervisor:
Fernando Gregori Pérez
Tutor académico:
Carlos Antonio Hernandéz Espinosa

Fecha de lectura: 20 de julio de 2016
Curso académico 2015/2016

Resumen

Este proyecto forma parte de un proyecto más grande nombrado Easygoband. Easygoband utiliza la tecnología NFC (NFC-Near-Communication) para ofrecer a sus clientes, un control de accesos, un control de stock, un control de aforo y un sistema de pago electrónico en eventos multitudinarios. Todos estos servicios los ofrece combinando una aplicación móvil para los usuarios que se utiliza para recargar dinero mediante el uso de la tarjeta de crédito, una aplicación móvil que va instalada en los dispositivos móviles que utilizan los operarios para poder gestionar el stock, el aforo, y el cobro electrónico (TPV) y una pulsera NFC que utilizan los usuarios para poder realizar todas sus compras dentro del recinto del evento sin necesidad de efectivo ni tarjetas crédito y para poder identificarse tanto en la entrada como en la salida del mismo.

Para poder ofrecer un producto único que contenga elementos más interactivos y amigables para los usuarios y un refuerzo al sistema de pagos, este proyecto tiene dos objetivos principales. 1: Desarrollar software para reforzar el sistema de pagos ofreciendo un punto físico donde el usuario pueda por sí mismo consultar de cuanto dinero dispone, qué pagos ha realizado, poder actualizar su pulsera NFC si ha realizado un recarga utilizando su aplicación móvil; 2: integrar las redes sociales. Para ello se han realizado dos aplicaciones:

La primera aplicación se trata de una interfaz, que muestra las publicaciones que realizan los asistentes a un evento multitudinario en el muro (público) del evento en Facebook, Twitter e Instagram en una pantalla de grandes dimensiones con una resolución FullHd(1920x1080p).

La segunda aplicación consiste en una interfaz interactiva diseñada para que el usuario pueda consultar su saldo, actualizar su pulsera NFC cuando se realice una recarga online mediante la aplicación móvil, sacar una fotografía con sus amigos y publicarla en Facebook utilizando solamente la pulsera NFC. Para poder publicar en Facebook el usuario deberá registrarse en su aplicación móvil con su cuenta de Facebook y permitir que la aplicación publique en su nombre para poder generar un token con permisos de publicación (al final del evento el token que se genera para poder publicar, caducará). Esta aplicación irá instalada en una Raspberry pi conectada a una cámara, un lector de NFC y una pantalla táctil. Todos estos componentes irán instalados en una estructura en forma de torreta que llamaremos Tótem.

Palabras clave

- **NFC:** Near-Field Communication es un conjunto de protocolos de comunicación que permiten a dos dispositivos electrónicos establecer una comunicación a una distancia de 4cm.
- **Redes sociales:** es una estructura social compuesta por un conjunto de actores (tales como individuos u organizaciones) que están relacionados de acuerdo a algún criterio (relación profesional, amistad, parentesco, etc.).
- **Evento multitudinario:** se trata de un evento de gran aforo al que asisten miles de personas.
- **Tótem:** es una estructura en forma de torreta que esta compuesta por una Raspberry pi, un pantalla táctil industrial, un lector NFC y un cámara (cam).
- **Token:** es un identificador o bandera que se utiliza para almacenar software que permite autenticarse. En nuestro caso en las redes sociales.
- **Endpoint:** es el módulo frontera de la aplicación que se encarga de realizar las peticiones a las APIs de las redes sociales e interpretarlas.
- **Amazon S3 bucket:** es un sistema de almacenamiento en la nube que sirve para subir tu información(fotografías, vídeos, documentos, etc):

Keywords

- **NFC:** Near-Field Communication (NFC) is a set of communication protocols that enable two electronic devices to establish communication by bringing them within about 4 cm of each other.
- **Social network:** it is a social structure made up of a set of social actors (such as individuals or organizations), sets of dyadic ties, and other social interactions between actors.
- **Mass-attendance event:** it is an event where thousands of people attend.
- **Totem:** it is a structure that looks like a torret and it is made of a Raspberry pi , a touchable industrial screen, a NFC reader and a cam.
- **Token:** It is a identificator or flag that stores software that allows for seamless authentication and password filling. In our case in social networks.
- **Endpoint:** it is the last applicaction module that has the fuction of making requests and interprete its to the social networks APIs
- **Amazon S3 bucket:** it is cloud storage for the Internet. To upload your data (photos, videos, documents etc.)

Índice general

Lista de figuras **7**

1. Introducción **9**

1.1. Contexto y motivación del proyecto 9

1.2. Objetivos del proyecto 9

1.3. Estructura de la memoria 10

2. Descripción del proyecto **11**

3. Planificación del proyecto **13**

3.1. Metodología 13

3.2. Planificación 15

3.3. Estimación de recursos y costes del proyecto 18

3.3.1. Recursos 18

3.3.2. Costes 24

3.4. Seguimiento del proyecto 25

4. Aplicación muro social **29**

4.1. Análisis y diseño del sistema 29

4.1.1. Análisis del sistema 29

4.1.2. Diseño de la arquitectura del sistema 34

4.1.3. Diseño de la interfaz 35
4.2. Implementación y pruebas 36
4.2.1. Detalles de implementación 36
4.2.2. Verificación y validación 44

5. Aplicación totem **45**
5.1. Análisis y diseño del sistema 45
5.1.1. Análisis del sistema 45
5.1.2. Diseño de la arquitectura del sistema 52
5.1.3. Diseño de la interfaz 53
5.2. Configuración de la pulsera NFC 55
5.3. Implementación y pruebas 55
5.3.1. Detalles de implementación 55
5.3.2. Verificación y validación 68
5.3.3. Configuración de la aplicación en la Raspberry pi 3 69

6. Conclusiones **71**

A. Manual gráfico de usuario de la aplicación del Tótem **75**
A.1. Consultar tu saldo: 76
A.2. Actualizar tu pulsera: 77
A.3. Sacarse una fotografía y compartirla en Facebook: 78

Índice de figuras

3.1. Diagrama TDD 13

3.2. Planificación inicial 15

3.3. Diagrama de Gantt 1 16

3.4. Diagrama de Gantt 2 16

3.5. Diagrama de Gantt 3 16

3.6. Diagrama de Gantt 4 17

4.1. Diagrama de flujo aplicación muro social 30

4.2. Diagrama de clases aplicación muro social 31

4.3. Diagrama de clases aplicación muro social ampliación 1 32

4.4. Diagrama de clases aplicación muro social aplicación 2 33

4.5. Diagrama base de datos cache local aplicación muro social 34

4.6. Interfaz aplicación muro social 35

4.7. Estructura del contenedor de las publicaciones 42

4.8. Estructura de la interfaz de la aplicación muro social 43

5.1. Diagrama de clases aplicación Tótem 46

5.2. Diagrama de clases aplicación Tótem 47

5.3. Diagrama de clases aplicación Tótem ampliado 1 48

5.4. Diagrama de clases aplicación Tótem ampliado 2 49

5.5. Diagrama de clases aplicación Tótem ampliado 3 50

5.6. Diagrama de clases aplicación Tótem ampliado 4 51

5.7. Diagrama base de datos de la cache local de la aplicación del Tótem 52

5.8. Pantallas aplicación Tótem 54

5.9. Estructura de la interfaz de la pantalla principal de la aplicación tótem 62

5.10. Estructura de la interfaz de la pantalla de consultar saldo de la aplicación tótem 63

5.11. Estructura de la interfaz de la pantalla lectura de la pulsera de la aplicación tótem 64

5.12. Estructura de la interfaz de la pantalla actualizar pulsera de la aplicación tótem 65

5.13. Estructura de la interfaz de la pantalla cuenta atrás de la aplicación tótem . . . 66

5.14. Estructura de la interfaz de la pantalla del *photocall* de la aplicación tótem . . . 67

A.1. Manual gráfico de usuario para consultar el saldo de tu pulsera 76

A.2. Manual gráfico de usuario para actualizar tu pulsera 77

A.3. Manual gráfico de usuario para sacarse una fotografía y compartirla en Facebook 78

Capítulo 1

Introducción

1.1. Contexto y motivación del proyecto

El proyecto NFC-RRSS se ha realizado en la empresa PaynoPain y forma parte de un proyecto más grande llamado EasygoBand. Este proyecto tiene como objetivo explotar la tecnología NFC para ofrecer a los usuarios una nueva experiencia en eventos, donde podrán utilizar su pulsera NFC para realizar pagos, gestionar sus compras, utilizar la pulsera como pase al evento y por último, conectar las redes sociales a esta experiencia de uso.

La motivación principal para realizar este proyecto era poder ofrecer un producto único y poder abrir nuevas e interesantes utilidades a esta tecnología que en este momento se encuentra en auge. Inicialmente el diseño de las aplicaciones está personalizado para un evento musical llamado MadCool que se ha realizado en Madrid en junio de este año. Durante este evento se ha podido ver en acción todas las funcionalidades realizadas en el proyecto y servirá como trampolín para poder aplicar esta nueva tendencia en más eventos culturales masivos en el futuro.

1.2. Objetivos del proyecto

El principal objetivo de este proyecto es la implementación de diferentes módulos en lenguaje Java que gestionen y realicen la comunicación entre el Tótem, la pulsera, el smartphone y las redes sociales.

Los objetivos principales se pueden desglosar en los siguientes:

- **Adaptar el Core de la aplicación:** consiste en crear módulos y modificar los ya existentes para gestionar los *Tokens* de Facebook que se generarán cuando el usuario se autentica con la aplicación móvil.

- **Añadir al *EndpointCore* de la aplicación el módulo SocialNetworks:** este módulo se encargará de realizar las peticiones a las textslAPIs de cada red social y gestionar su respuesta para ser utilizada en las aplicaciones finales.
- **Crear la aplicación *Social Wall*:** esta aplicación se desarrollará en *JavaFx* [6] y consistirá en una interfaz para una pantalla gigante donde se retransmitirían todas las publicaciones que los usuarios realicen en las páginas del evento y en las redes sociales (Facebook, Twitter e Instagram).
- **Crear la aplicación que irá instalada en los *Tótem*:** esta aplicación ofrecerá tres funcionalidades a los usuarios: consultar su saldo y sus transacciones realizadas con la pulsera, actualizar la información de la pulsera, hacerse una fotografía y publicarla en Facebook.
- **Configurar e instalar las aplicaciones en la *Raspberry pi 3*:** este objetivo consiste en lograr que las aplicaciones se ejecuten perfectamente en un sistema empotrado ARM (Advanced RISC Machine) y conseguir optimizar sus recursos.

1.3. Estructura de la memoria

La estructura de la memoria se dividirá en tres bloques principales y una conclusión acompañada con un anexo que contendrá un manual de uso de la aplicación del Totem y la instalación de los componentes hardware. Los tres bloques principales serán los siguientes:

- **Planificación del proyecto:** en este capítulo se expondrá cómo se ha planificado el proyecto y se detallará cómo se ha adaptado la planificación durante el desarrollo del mismo.
- **Análisis y diseño del sistema:** en este capítulo se explicará detalladamente qué componentes se han utilizado en el proyecto y cuál ha sido el diseño realizado para las interfaces de usuario.
- **Implementación y pruebas** : finalmente en este capítulo se entrará en detalle en la implementación y se expondrán las pruebas y la validación de éstas.

Capítulo 2

Descripción del proyecto

El sistema a desarrollar consiste en enlazar un dispositivo NFC con las redes sociales con mayor comunidad de usuarios activos actualmente. Consistirá en diferentes módulos que ofrecerán al usuario una experiencia completa de conexión. Por una parte se desarrollará el módulo correspondiente para la aplicación de usuario que se podrá instalar en su smartphone. Esta aplicación solicitará el acceso a las diferentes plataformas y vinculará el dispositivo NFC personal del usuario (la pulsera NFC) con dicho acceso. El usuario luego podrá acceder a un punto de información multimedia *(tótem)* a través del cual podrá interaccionar con las plataformas previamente vinculadas. Esta interacción consistirá en la toma de una fotografía que posteriormente se subirá al muro de Facebook del usuario, consultar el saldo disponible almacenado en su pulsera NFC y actualizar su pulsera NFC con el nuevo saldo en el caso de que el usuario haya realizado una recarga mediante la aplicación móvil.

También se desarrollará un módulo que extrae las publicaciones en las distintas plataformas y las muestra de forma unificada en una pantalla de grandes dimensiones.

El sistema permitirá solicitar el acceso a diferentes redes sociales *(Facebook,Twitter,Instagram)*, para poder extraer diferente contenido social *(fotos,comentarios,etc)* del evento publicitado en las diferentes redes sociales y poder publicar en el muro de Facebook del usuario las fotografías tomadas en el Tótem.

El sistema ha de gestionar la información extraída e introducida en las redes sociales mediante peticiones GET y POST con el uso de parámetros de autenticación (Tokens) para cada tipo de petición a los diferentes servidores de las redes sociales soportadas por el sistema. Por otro lado debe gestionar y organizar tal información para el posterior almacenamiento y uso de ésta.

Un sistema como éste ofrecerá a los usuarios una experiencia diferente e innovadora de interacción con las redes sociales gracias a la tecnología NFC. En concreto en este proyecto se pretende utilizar pulseras con un dispositivo NFC que se utilizarán para interaccionar con los diferentes *Tótem* que estarán dispersos por el recinto del evento.

Capítulo 3

Planificación del proyecto

3.1. Metodología

La metodología de trabajo que se emplea en la empresa consiste en una práctica de programación denominada TDD o Test-Driven Development (desarrollo dirigido por tests) que consiste en escribir primero las pruebas (generalmente unitarias), después escribir el código fuente, seguidamente pasar las pruebas satisfactoriamente y, por último, refactorizar el código escrito.

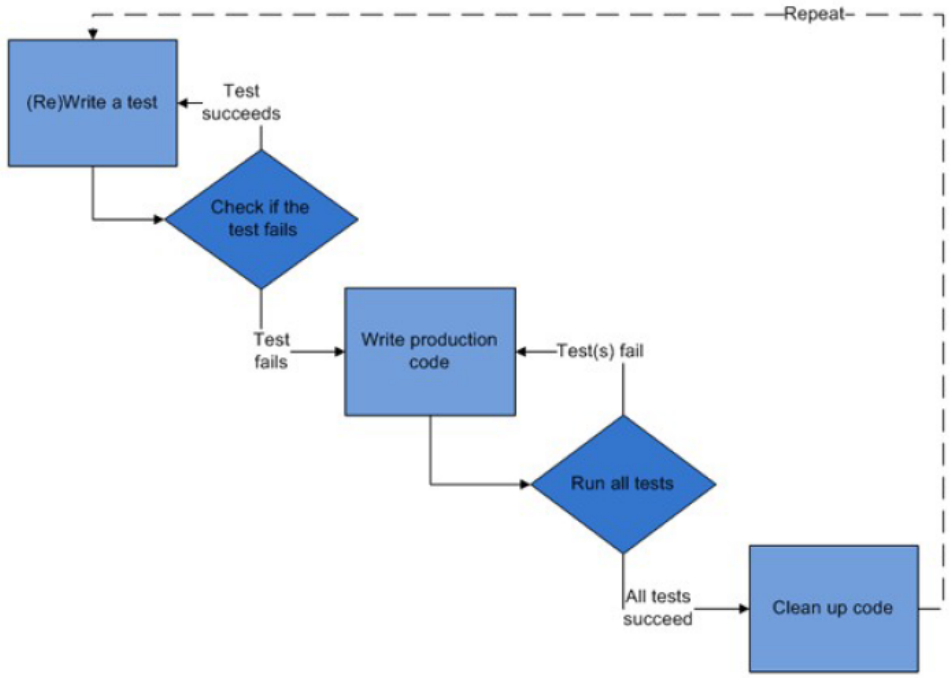


Figura 3.1: Diagrama TDD

Por otra parte la empresa, para organizar y planificar cada proyecto, utiliza una interfaz web propia para asignar tareas a cada trabajador. En la interfaz cada trabajador puede consultar qué tareas tiene, cuando hay que hacerlas y una pequeña descripción de éstas. Así aunque haya una planificación inicial, ésta puede alterarse y comunicarse a los trabajadores rápidamente.

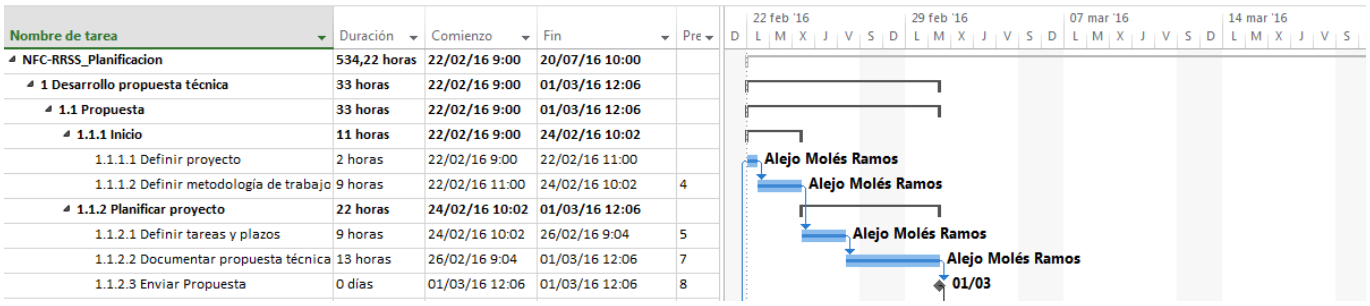
3.2. Planificación

La planificación inicial del proyecto se estructuró para realizar las tareas reflejadas en la tabla 3.2.

La variación más importante con respecto a la planificación inicial es que, la planificación de las etapas de planificación y desarrollo en las aplicaciones inicialmente están unidas y durante la estancia se dividieron en dos proyectos diferentes. Primero se realizó la aplicación del muro social y a continuación se realizó la aplicación del Tótem.

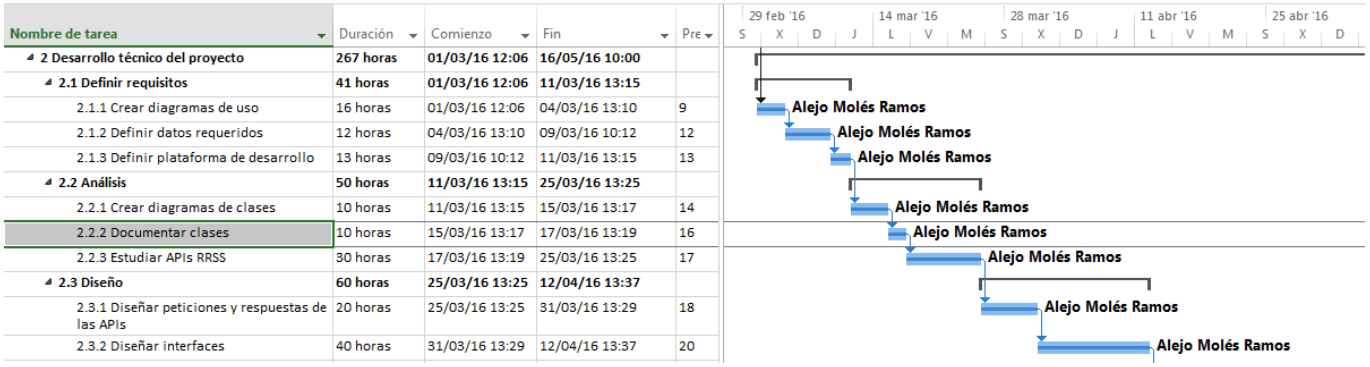
	NFC-RRSS_Planificacion	Duración	Comienzo	Fin	Predece	Nombres de los recursos
0	**NFC-RRSS_Planificacion**	**534,22 horas**	**22/02/16 9:00**	**20/07/16 10:00**		
1	**Desarrollo propuesta técnica**	**33 horas**	**22/02/16 9:00**	**01/03/16 12:06**		
2	**Propuesta**	**33 horas**	**22/02/16 9:00**	**01/03/16 12:06**		
3	**Inicio**	**11 horas**	**22/02/16 9:00**	**24/02/16 10:02**		
4	Definir proyecto	2 horas	22/02/16 9:00	22/02/16 11:00		Alejo Molés Ramos
5	Definir metodología de trabajo	9 horas	22/02/16 11:00	24/02/16 10:02	4	Alejo Molés Ramos
6	**Planificar proyecto**	**22 horas**	**24/02/16 10:02**	**01/03/16 12:06**		
7	Definir tareas y plazos	9 horas	24/02/16 10:02	26/02/16 9:04	5	Alejo Molés Ramos
8	Documentar propuesta técnica	13 horas	26/02/16 9:04	01/03/16 12:06	7	Alejo Molés Ramos
9	Enviar Propuesta	0 días	01/03/16 12:06	01/03/16 12:06	8	Alejo Molés Ramos
10	**Desarrollo técnico del proyecto**	**267 horas**	**01/03/16 12:06**	**16/05/16 10:00**		
11	**Definir requisitos**	**41 horas**	**01/03/16 12:06**	**11/03/16 13:15**		
12	Crear diagramas de uso	16 horas	01/03/16 12:06	04/03/16 13:10	9	Alejo Molés Ramos
13	Definir datos requeridos	12 horas	04/03/16 13:10	09/03/16 10:12	12	Alejo Molés Ramos
14	Definir plataforma de desarrollo	13 horas	09/03/16 10:12	11/03/16 13:15	13	Alejo Molés Ramos
15	**Análisis**	**50 horas**	**11/03/16 13:15**	**25/03/16 13:25**		
16	Crear diagramas de clases	10 horas	11/03/16 13:15	15/03/16 13:17	14	Alejo Molés Ramos
17	Documentar clases	10 horas	15/03/16 13:17	17/03/16 13:19	16	Alejo Molés Ramos
18	Estudiar APIs RRSS	30 horas	17/03/16 13:19	25/03/16 13:25	17	Alejo Molés Ramos
19	**Diseño**	**60 horas**	**25/03/16 13:25**	**12/04/16 13:37**		
20	Diseñar peticiones y respuestas de las APIs	20 horas	25/03/16 13:25	31/03/16 13:29	18	Alejo Molés Ramos
21	Diseñar interfaces	40 horas	31/03/16 13:29	12/04/16 13:37	20	Alejo Molés Ramos
22	**Desarrollo del producto**	**110 horas**	**12/04/16 13:37**	**12/05/16 13:59**		
23	Escribir test	13 horas	12/04/16 13:37	15/04/16 11:39	21	Alejo Molés Ramos
24	Escribir código	85 horas	15/04/16 11:39	10/05/16 11:56	23	Alejo Molés Ramos
25	Ejecutar test de validación	2 horas	10/05/16 11:56	10/05/16 13:57	24	Alejo Molés Ramos
26	Refinar código	10 horas	10/05/16 13:57	12/05/16 13:59	25	Alejo Molés Ramos
27	**Pruebas de funcionamiento**	**6 horas**	**12/05/16 13:59**	**16/05/16 10:00**		
28	Ejecución en entorno de desarrollo	1 hora	12/05/16 13:59	13/05/16 9:59	26	Alejo Molés Ramos
29	Validación en entorno de desarrollo	2 horas	13/05/16 9:59	13/05/16 11:59	28	Alejo Molés Ramos
30	Ejecución en entorno final	1 hora	13/05/16 11:59	13/05/16 12:59	29	Alejo Molés Ramos
31	Validación en entorno final	2 horas	13/05/16 12:59	16/05/16 10:00	30	Alejo Molés Ramos
32	Entrega producto	0 horas	16/05/16 10:00	16/05/16 10:00	31	
33	**Documentación y presentación TFG**	**486,37 horas**	**04/03/16 12:00**	**20/07/16 10:00**		
34	**Redacción informes quincenales**	**281,05 horas**	**04/03/16 12:00**	**23/05/16 20:00**		
35	Informe quincenal 1	2 horas	04/03/16 12:00	04/03/16 14:01	4CC+15	Alejo Molés Ramos
36	Informe quincenal 2	2 horas	18/03/16 9:00	18/03/16 11:00	35FC+1	Alejo Molés Ramos
37	Informe quincenal 3	2 horas	31/03/16 12:00	31/03/16 14:01	36CC+1	Alejo Molés Ramos
38	Informe quincenal 4	2 horas	13/04/16 17:00	13/04/16 19:00	37CC+1	Alejo Molés Ramos
39	Informe quincenal 5	2 horas	27/04/16 10:00	27/04/16 12:00	38CC+1	Alejo Molés Ramos
40	Informe quincenal 6	2 horas	10/05/16 15:00	10/05/16 17:00	39CC+1	Alejo Molés Ramos
41	Informe quincenal 7	2 horas	23/05/16 18:00	23/05/16 20:00	40CC+1	Alejo Molés Ramos
42	Redacción memoria	115 horas	16/05/16 10:00	16/06/16 10:23	32	Alejo Molés Ramos
43	Entrega memoria técnica	0 días	05/07/16 9:00	05/07/16 9:00		Alejo Molés Ramos
44	Preparación presentación oral	30 horas	05/07/16 9:00	13/07/16 9:06	43	Alejo Molés Ramos
45	Presentación oral	1 hora	20/07/16 9:00	20/07/16 10:00		Alejo Molés Ramos

Figura 3.2: Planificación inicial



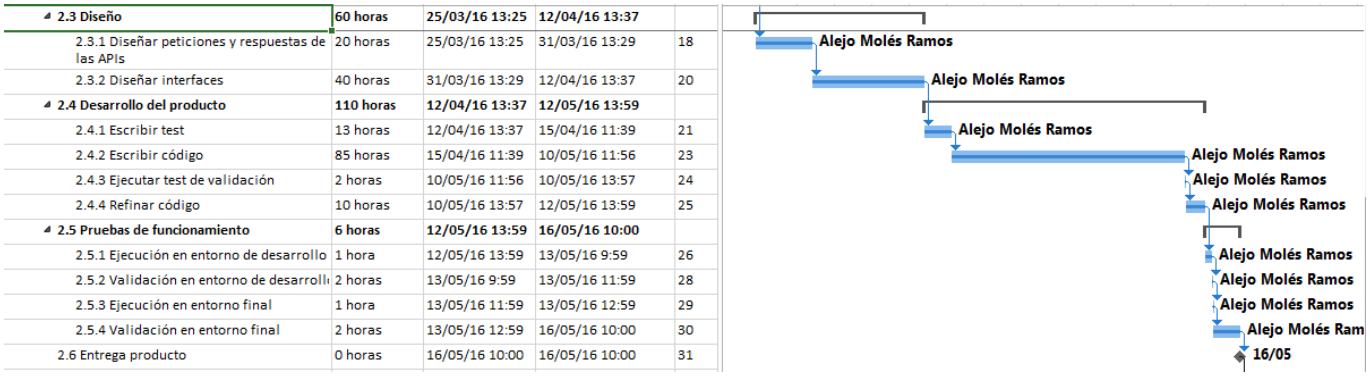
Nombre de tarea	Duración	Comienzo	Fin	Pre
NFC-RRSS_Planificacion	**534,22 horas**	**22/02/16 9:00**	**20/07/16 10:00**	
1 Desarrollo propuesta técnica	**33 horas**	**22/02/16 9:00**	**01/03/16 12:06**	
1.1 Propuesta	**33 horas**	**22/02/16 9:00**	**01/03/16 12:06**	
1.1.1 Inicio	**11 horas**	**22/02/16 9:00**	**24/02/16 10:02**	
1.1.1.1 Definir proyecto	2 horas	22/02/16 9:00	22/02/16 11:00	
1.1.1.2 Definir metodología de trabajo	9 horas	22/02/16 11:00	24/02/16 10:02	4
1.1.2 Planificar proyecto	**22 horas**	**24/02/16 10:02**	**01/03/16 12:06**	
1.1.2.1 Definir tareas y plazos	9 horas	24/02/16 10:02	26/02/16 9:04	5
1.1.2.2 Documentar propuesta técnica	13 horas	26/02/16 9:04	01/03/16 12:06	7
1.1.2.3 Enviar Propuesta	0 días	01/03/16 12:06	01/03/16 12:06	8

Figura 3.3: Diagrama de Gantt 1



Nombre de tarea	Duración	Comienzo	Fin	Pre
2 Desarrollo técnico del proyecto	**267 horas**	**01/03/16 12:06**	**16/05/16 10:00**	
2.1 Definir requisitos	**41 horas**	**01/03/16 12:06**	**11/03/16 13:15**	
2.1.1 Crear diagramas de uso	16 horas	01/03/16 12:06	04/03/16 13:10	9
2.1.2 Definir datos requeridos	12 horas	04/03/16 13:10	09/03/16 10:12	12
2.1.3 Definir plataforma de desarrollo	13 horas	09/03/16 10:12	11/03/16 13:15	13
2.2 Análisis	**50 horas**	**11/03/16 13:15**	**25/03/16 13:25**	
2.2.1 Crear diagramas de clases	10 horas	11/03/16 13:15	15/03/16 13:17	14
2.2.2 Documentar clases	10 horas	15/03/16 13:17	17/03/16 13:19	16
2.2.3 Estudiar APIs RRSS	30 horas	17/03/16 13:19	25/03/16 13:25	17
2.3 Diseño	**60 horas**	**25/03/16 13:25**	**12/04/16 13:37**	
2.3.1 Diseñar peticiones y respuestas de las APIs	20 horas	25/03/16 13:25	31/03/16 13:29	18
2.3.2 Diseñar interfaces	40 horas	31/03/16 13:29	12/04/16 13:37	20

Figura 3.4: Diagrama de Gantt 2



2.3 Diseño	**60 horas**	**25/03/16 13:25**	**12/04/16 13:37**	
2.3.1 Diseñar peticiones y respuestas de las APIs	20 horas	25/03/16 13:25	31/03/16 13:29	18
2.3.2 Diseñar interfaces	40 horas	31/03/16 13:29	12/04/16 13:37	20
2.4 Desarrollo del producto	**110 horas**	**12/04/16 13:37**	**12/05/16 13:59**	
2.4.1 Escribir test	13 horas	12/04/16 13:37	15/04/16 11:39	21
2.4.2 Escribir código	85 horas	15/04/16 11:39	10/05/16 11:56	23
2.4.3 Ejecutar test de validación	2 horas	10/05/16 11:56	10/05/16 13:57	24
2.4.4 Refinar código	10 horas	10/05/16 13:57	12/05/16 13:59	25
2.5 Pruebas de funcionamiento	**6 horas**	**12/05/16 13:59**	**16/05/16 10:00**	
2.5.1 Ejecución en entorno de desarrollo	1 hora	12/05/16 13:59	13/05/16 9:59	26
2.5.2 Validación en entorno de desarroll	2 horas	13/05/16 9:59	13/05/16 11:59	28
2.5.3 Ejecución en entorno final	1 hora	13/05/16 11:59	13/05/16 12:59	29
2.5.4 Validación en entorno final	2 horas	13/05/16 12:59	16/05/16 10:00	30
2.6 Entrega producto	0 horas	16/05/16 10:00	16/05/16 10:00	31

Figura 3.5: Diagrama de Gantt 3

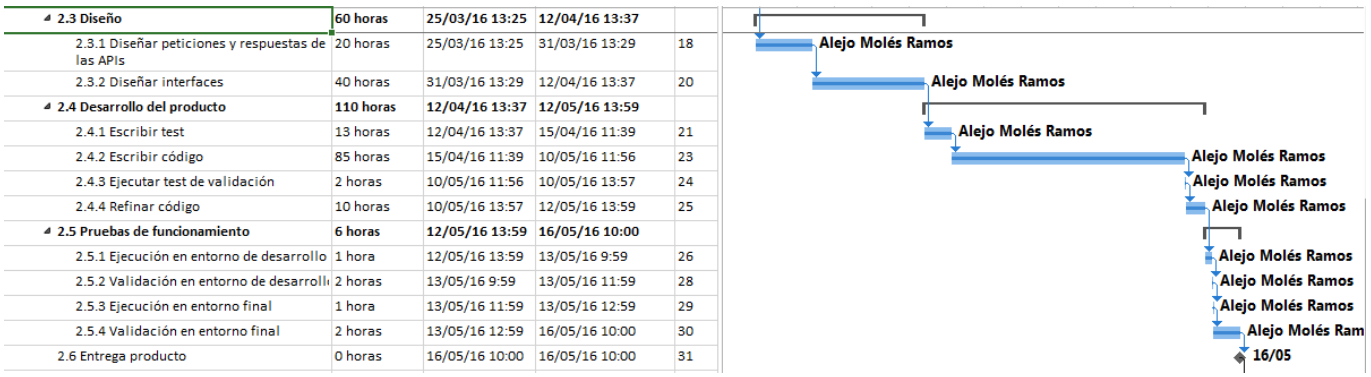


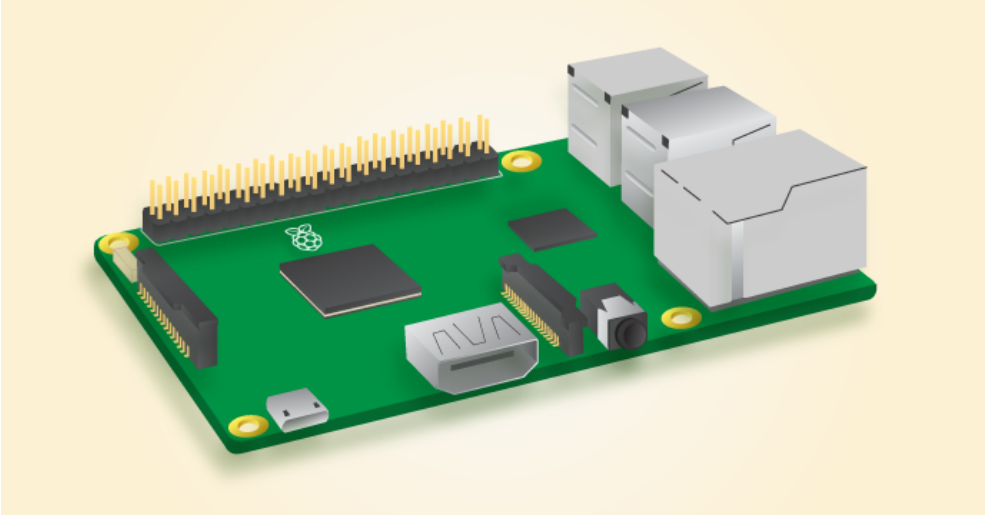
Figura 3.6: Diagrama de Gantt 4

3.3. Estimación de recursos y costes del proyecto

3.3.1. Recursos

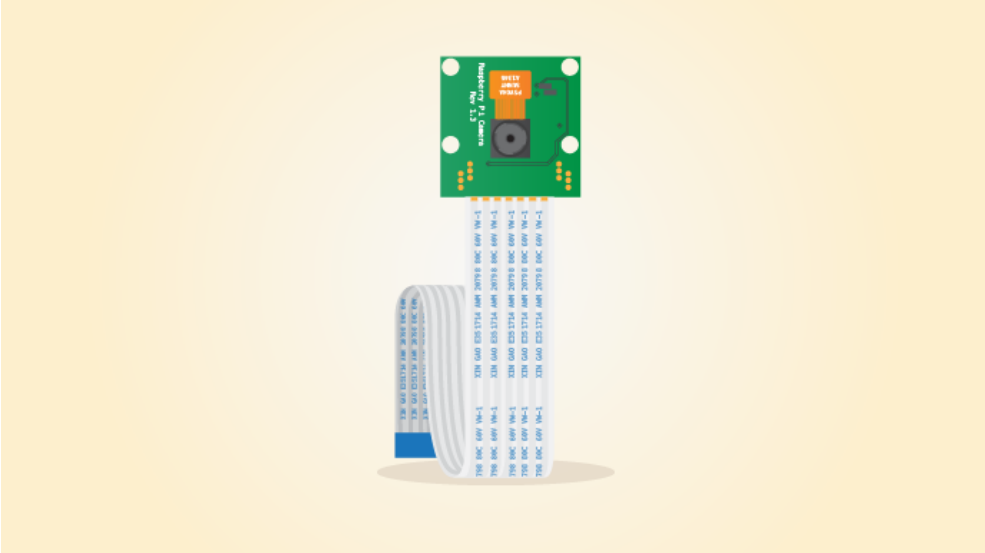
Hardware

- **Raspberry pi 3:**



- A 1.2GHz 64-bit quad-core ARMv8 CPU
- 802.11n Wireless LAN
- Bluetooth 4.1
- Bluetooth Low Energy (BLE)
- 1GB RAM
- 4 USB ports
- 40 GPIO pins
- Full HDMI port
- Ethernet port
- Combined 3.5mm audio jack and composite video
- Camera interface (CSI)
- Display interface (DSI)
- Micro SD card slot (now push-pull rather than push-push)
- VideoCore IV 3D graphics core

- **Módulo cámara Web cam para Raspberry:**

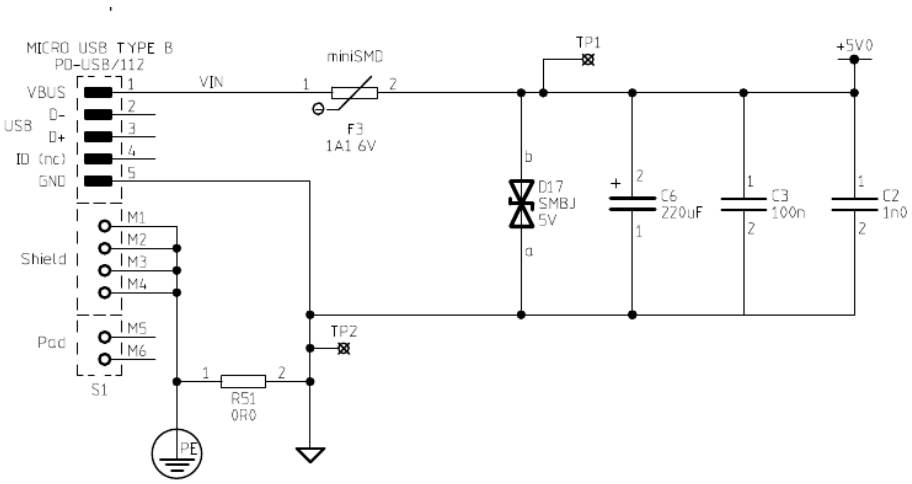


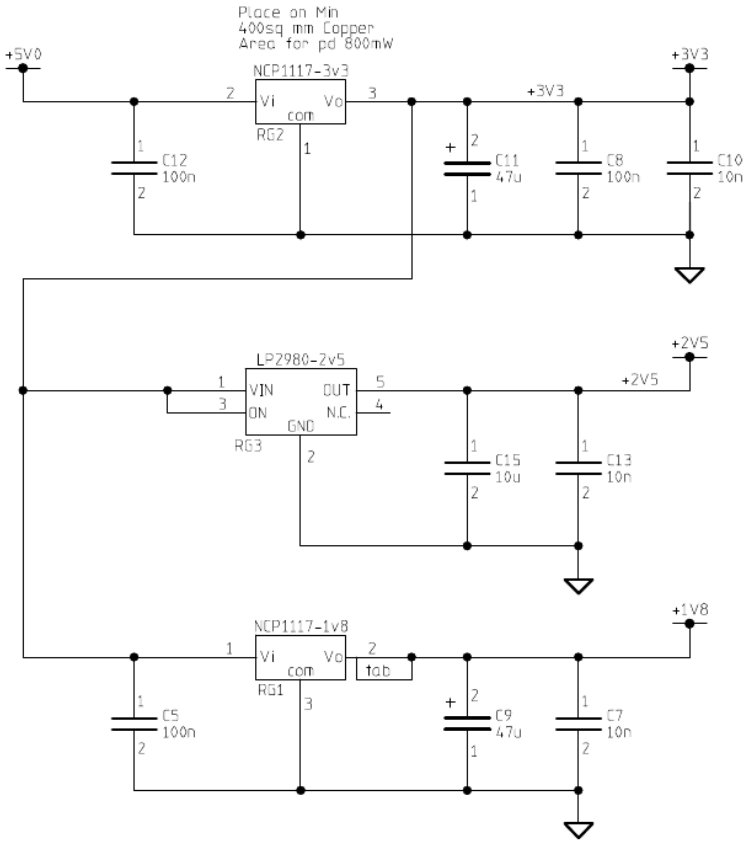
Se trata de una cámara de vídeo de alta definición que tiene 5 megapixels y soporta las siguientes resoluciones de vídeo: 1080p30, 720p60 y VGA90. Aunque es una cámara de vídeo también es capaz de sacar fotos de alta resolución.

- **Samsung 32GB EVO Class 10 microSD Card :**

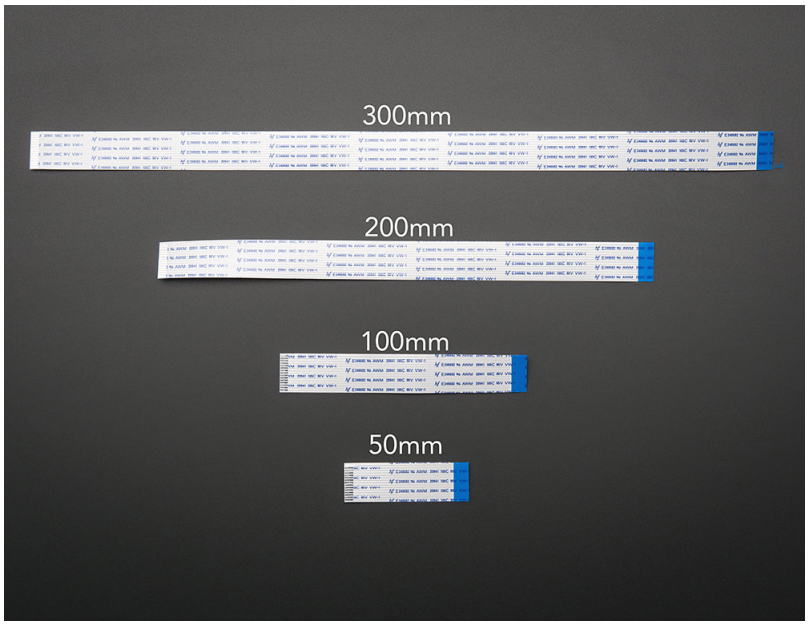
- Dimensiones: 15x11x1 (mm).
- Velocidad de transferencia hasta 48 MB/s.
- Capacidad: 32 GB.
- Interfaz: UHS-I, compatible con interfaz HS

- **PSU Raspberry:** Se trata de una fuente de alimentación con un conector microUSB de 5V y 1.2A(1200mA). Se eligió esta fuente de alimentación ya que aunque la Raspberry consume entre 700-1000mA había que conectar el puerto HDMI que consume 50mA y la cámara que consume 250mA. A continuación se muestra el esquema de la fuente de alimentación.





- **Flex cable module camera Raspberry:** Cuando se compró el modulo de la cámara llevaba un cable de 10cm pero era necesario un cable de mayores dimensiones para la instalación de los componentes en el Tótem. Así que se determinó comprar cables de 30cm

+++ 

- **Pulsera NFC:**

- **Totem:** El Tótem lo ha fabricado una empresa externa.

- **Pantalla gigante:** Esta pantalla no entra en el presupuesto ya que la aporta la organización del evento. Aunque habría que acordar con la organización del evento la obtención de ésta.

Software

- **Java 8:** Todas las aplicaciones y los módulos están escritas en Java.
- **JavaFx:** [6] Las interfaces de las aplicaciones están implementadas en JavaFx.
- **Mysql:** Las aplicaciones disponen de una cache local donde se almacenan las respuestas de los servidores de las redes sociales, para poder utilizar la información que almacena en caso de un fallo en la conexión en la red.
- **API Facebook, Twitter e Instagram:** las APIs se han utilizado para poder extraer y publicar información en las redes sociales.
- **Bash script:** se ha utilizado para realizar los scripts de auto ejecución de las aplicaciones en la Raspberry.
- **JSON:** las respuestas de los servidores de las redes sociales viene en formato JSON.
- **JRPiCam:** [5] JRPiCam se trata de una API escrita en Java que permite poder gestionar la cámara desde la aplicación.

3.3.2. Costes

A continuación se muestra una tabla que recoge los precios desglosados por las diferentes tareas realizadas en el proyecto:

NFC-RRSS_Planificacion_20160727

Id	Nombre de tarea	Duración	Costo
0	**NFC-RRSS_Planificacion_20160727**	**534,22 horas**	**€2.700,00**
1	**Desarrollo propuesta técnica**	**33 horas**	**€297,00**
2	**Propuesta**	**33 horas**	**€297,00**
3	**Inicio**	**11 horas**	**€99,00**
4	Definir proyecto	2 horas	€18,00
5	Definir metodología de trabajo	9 horas	€81,00
6	**Planificar proyecto**	**22 horas**	**€198,00**
7	Definir tareas y plazos	9 horas	€81,00
8	Documentar propuesta técnica	13 horas	€117,00
9	Enviar Propuesta	0 días	€0,00
10	**Desarrollo técnico del proyecto**	**267 horas**	**€2.403,00**
11	**Definir requisitos**	**41 horas**	**€369,00**
12	Crear diagramas de uso	16 horas	€144,00
13	Definir datos requeridos	12 horas	€108,00
14	Definir plataforma de desarrollo	13 horas	€117,00
15	**Análisis**	**50 horas**	**€450,00**
16	Crear diagramas de clases	10 horas	€90,00
17	Documentar clases	10 horas	€90,00
18	Estudiar APIs RRSS	30 horas	€270,00
19	**Diseño**	**60 horas**	**€540,00**
20	Diseñar peticiones y respuestas de las APIs	20 horas	€180,00
21	Diseñar interfaces	40 horas	€360,00
22	**Desarrollo del producto**	**110 horas**	**€990,00**
23	Escribir test	13 horas	€117,00
24	Escribir código	85 horas	€765,00
25	Ejecutar test de validación	2 horas	€18,00
26	Refinar código	10 horas	€90,00
27	**Pruebas de funcionamiento**	**6 horas**	**€54,00**
28	Ejecución en entorno de desarrollo	1 hora	€9,00
29	Validación en entorno de desarrollo	2 horas	€18,00
30	Ejecución en entorno final	1 hora	€9,00
31	Validación en entorno final	2 horas	€18,00
32	Entrega producto	0 horas	€0,00
33	**Documentación y presentación TFG**	**486,37 horas**	**€0,00**
34	**Redacción informes quincenales**	**281,05 horas**	**€0,00**
35	Informe quincenal 1	0 horas	€0,00
36	Informe quincenal 2	0 horas	€0,00
37	Informe quincenal 3	0 horas	€0,00
38	Informe quincenal 4	0 horas	€0,00
39	Informe quincenal 5	0 horas	€0,00
40	Informe quincenal 6	0 horas	€0,00
41	Informe quincenal 7	0 horas	€0,00
42	Redacción memoria	0 horas	€0,00
43	Entrega memoria técnica	0 días	€0,00
44	Preparación presentación oral	0 horas	€0,00
45	Presentación oral	0 horas	€0,00

3.4. Seguimiento del proyecto

Como se ha comentado anteriormente, se ha utilizado una aplicación web para asignar las tareas. La metodología seguida a lo largo de la estancia en prácticas ha sido la siguiente: establecer la tareas, desarrollarlas y validar su correcto funcionamiento.

Cuando surgía algún contratiempo, por ejemplo un cambio en la lógica o un retraso, se creaba una nueva tarea para solucionarlo o se extendía el periodo de la actual. En este aspecto la empresa era muy flexible.

Por otro lado si se tenía que realizar otra tarea que se había planificado realizar más tarde por alguna razón externa (configurar Raspberries para mostrarlas al cliente) se alteraba la fecha de las tareas.

Es decir, en la empresa se aplica una planificación flexible para poder adaptarse a cada momento.

Como se puede ver en la lista de tareas realizadas se han distribuido con objetivos muy concretos para cada componente/módulo que iba a formar las aplicaciones finales. Este desglose tan detallado no se podía apreciar en la planificación inicial y expone una variación en las tareas planificadas inicialmente ya que aunque al principio sí que se hizo un investigación para poder confirmar la viabilidad del proyecto, posteriormente cada proceso de investigación e implementación de cada módulo de la aplicación se hizo por separado. Una vez realizados todos los módulos se juntaron en una aplicación común y entonces se detallaron los diagramas de flujo de cada una de las aplicaciones, su diseño estético y su posterior implementación.

A continuación se muestra la lista de tareas que realicé durante el periodo de prácticas:

Tasks Report

easyGOband — easyGOband

Generated: 19 May 2016 15:58

Inbox

Completed Tasks

Task	Start Date	Date Due	Responsible	Assigned By	Priority	Progress	Status	Time	Billable	Estimated
Vinculación spotify			Alejo M.	Christian B.		100%	Completed 18 May (2016) by Alejo M.	0	0	0
								None	None	None

Totem

Completed Tasks

Task	Start Date	Date Due	Responsible	Assigned By	Priority	Progress	Status	Time	Billable	Estimated
Estudiar API facebook y resumir todas sus funcionalidades		06 May (2016)	Alejo M.	Christian B.		100%	Completed 04 Apr (2016) by Christian B.	0	0	0
Aplicación de FEED para redes sociales (Facebook, Twitter, Instagram)		06 May (2016)	Alejo M.	Christian B.		100%	Completed 04 Apr (2016) by Christian B.	0	0	0
• UseCase FetchPosts	25 Feb (2016)	25 Feb (2016)	Alejo M.	Alejo M.		100%	Completed 26 Feb (2016) by Alejo M.	0	0	0
• FetchPostsFacebook		06 May (2016)	Alejo M.	Alejo M.		100%	Completed 01 Mar (2016) by Alejo M.	0	0	0

1. **Alejo Moles Ramos** February 22, 2016 at 09:56

- Login via facebook (Token de acceso, permisos).
- Compartir contenido via facebook (imagenes, videos, links,comentarios).
- Botones facebook (like, compartir, enviar).
- Monitorización de eventos.
- Compartir contenido via mensage.
- Conectar con aplicaciones propias.

• FetchPostsFacebook		06 May (2016)	Alejo M.	Alejo M.		100%	Completed 01 Mar (2016) by Alejo M.	0	0	0
• GetFacebookPosts		06 May (2016)	Alejo M.	Alejo M.		100%	Completed 01 Mar (2016) by Alejo M.	0	0	0
• GetTwiitterPosts		06 May (2016)	Alejo M.	Alejo M.		100%	Completed 01 Mar (2016) by Alejo M.	0	0	0
• GetInstagramPosts	04 Mar (2016)	07 Mar (2016)	Alejo M.	Alejo M.		100%	Completed 01 Mar (2016) by Alejo M.	0	0	0
• GettwitterPOsts		03 Mar (2016)	Alejo M.	Alejo M.		100%	Completed 04 Mar (2016) by Alejo M.	0	0	0
• GetFacebookPosts	22 Feb (2016)	01 Mar (2016)	Alejo M.	Alejo M.		100%	Completed 01 Mar (2016) by Alejo M.	0	0	0
• GetFacebookPosts	22 Feb (2016)	01 Mar (2016)	Alejo M.	Alejo M.		100%	Completed 01 Mar (2016) by Alejo M.	0	0	0
• GetInstagramPosts	04 Mar (2016)	07 Mar (2016)	Alejo M.	Alejo M.		100%	Completed 04 Mar (2016) by Alejo M.	0	0	0
• Añadir likes/tweets a los parametros	04 Mar (2016)	04 Mar (2016)	Alejo M.	Alejo M.		100%	Completed 04 Mar (2016) by Alejo M.	0	0	0
• Revisión	07 Mar (2016)	09 Mar (2016)	Alejo M.	Alejo M.		100%	Completed 10 Mar (2016) by Alejo M.	0	0	0
Aplicacion Post para redes sociales (Facebook)	30 Mar (2016)	18 May (2016)	Alejo M.	Alejo M.		100%	Completed 13 Apr (2016) by Christian B.	0	0	0
• EndPoint Post to Facebook	31 Mar (2016)	18 May (2016)	Alejo M.	Alejo M.		100%	Completed 06 Apr (2016) by Alejo M.	0	0	0
• App Photocall	30 Mar (2016)	18 May (2016)	Alejo M.	Alejo M.		100%	Completed 08 Apr (2016) by Alejo M.	0	0	0
• Tag persona en la foto o enviarle la foto via e-mail	30 Mar (2016)	18 May (2016)	Alejo M.	Alejo M.		100%	Completed 13 Apr (2016) by Christian B.	0	0	0
Interfez pantalla de los posts	07 Mar (2016)	25 Mar (2016)	Alejo M.	Alejo M.		100%	Completed 04 Apr (2016) by Christian B.	0	0	0
• Implementar esqueleto de la pantalla	10 Mar (2016)	10 Mar (2016)	Alejo M.	Alejo M.		100%	Completed 10 Mar (2016) by Alejo M.	0	0	0
• Implementar Controlador	11 Mar (2016)	14 Mar (2016)	Alejo M.	Alejo M.		100%	Completed 17 Mar (2016) by Alejo M.	0	0	0
• Depuración	15 Mar (2016)	25 Mar (2016)	Alejo M.	Alejo M.		100%	Completed 30 Mar (2016) by Alejo M.	0	0	0
• Display Posts	17 Mar (2016)	25 Mar (2016)	Alejo M.	Alejo M.		100%	Completed 30 Mar (2016) by Alejo M.	0	0	0
RapsBerry setUp	08 Apr (2016)	18 May (2016)	Alejo M.	Alejo M.		100%	Completed 13 Apr (2016) by Christian B.	0	0	0

Pantalla de tomar foto y subirla a facebook	13 Apr (2016)	19 Apr (2016)	Alejo M.	Christian B.		100%	Completed 28 Apr (2016) by Alejo M.	0	0	0
Pantalla de inicio En esta pantalla se mostrará el menú de acciones: - Consulta de saldo - Actualizar saldo - Tomar y subir foto	19 Apr (2016)	19 Apr (2016)	Alejo M.	Christian B.		100%	Completed 25 Apr (2016) by Alejo M.	0	0	0
Pantalla de consulta de saldo Saldo y listado de transacciones	20 Apr (2016)	22 Apr (2016)	Alejo M.	Christian B.		100%	Completed 26 Apr (2016) by Alejo M.	0	0	0
Pantalla de actualizar saldo Al usuario se le indica que pase la pulsera para que se actualice el saldo	25 Apr (2016)	25 Apr (2016)	Alejo M.	Christian B.		100%	Completed 28 Apr (2016) by Alejo M.	0	0	0
En la pantalla inicial los textos no están correctamente alineados	03 May (2016)	03 May (2016)	Alejo M.	Christian B.		100%	Completed 03 May (2016) by Alejo M.	0	0	0
En la pantalla intermedia de consultar saldo, el texto no está bien alineado	03 May (2016)	03 May (2016)	Alejo M.	Christian B.		100%	Completed 03 May (2016) by Alejo M.	0	0	0
Las transiciones entre pantalla (el fadeout,fadein) hay que eliminarlo.	03 May (2016)	03 May (2016)	Alejo M.	Christian B.		100%	Completed 03 May (2016) by Alejo M.	0	0	0
El cargando en la pantalla intermedia de consulta de salto desaparece demasiado pronto, no espera a que se hayan descargado las transacciones	03 May (2016)	03 May (2016)	Alejo M.	Christian B.		100%	Completed 03 May (2016) by Alejo M.	0	0	0
En la pantalla intermedia de actualizar saldo no debería indicar cuantas transacciones pendientes quedan	04 May (2016)	04 May (2016)	Alejo M.	Christian B.		100%	Completed 03 May (2016) by Alejo M.	0	0	0
La pantalla de los detalles del saldo no se parece a la diseñada	04 May (2016)	05 May (2016)	Alejo M.	Christian B.		100%	Completed 04 May (2016) by Alejo M.	0	0	0
La pantalla de detalles transaccionales no se parece a la diseñada y sólo debe abrirse para las transacciones que contengan cart items y sean de tipo order. Además el modal de los detalles hace algo raro y tienes que pinchar dos o tres veces antes de que se cierre (sobre el tick amarillo).	04 May (2016)	06 May (2016)	Alejo M.	Christian B.		100%	Completed 05 May (2016) by Alejo M.	0	0	0
Preparar vídeo para aprobación de Facebook	06 May (2016)	10 May (2016)	Alejo M.	Christian B.		100%	Completed 17 May (2016) by Christian B.	0	0	0
Nueva lógica de subir imágenes a Facebook		06 May (2016)	Alejo M.	Christian B.		100%	Completed 18 May (2016) by Alejo M.	0	0	0
								None	**None**	**None**

Generated for Alejo Moles Ramos at 15:58 19/05/2016

Capítulo 4

Aplicación muro social

4.1. Análisis y diseño del sistema

4.1.1. Análisis del sistema

Requisitos

- Conseguir extraer la información de las redes sociales utilizado sus respectivas APIs.
- Desarrollar un diseño corporativo que se adapte a la imagen de la empresa.
- Conseguir que la aplicación siga ejecutándose con normalidad si se pierde la conexión a la red.
- La aplicación debe desarrollarse con JavaFx. [6]
- Se debe conseguir que se ejecute de forma óptima en el dispositivo Raspberry pi 3 con arquitectura ARM.
- La interfaz de la aplicación tiene que tener una resolución fullHD(1920p1082).

Diagrama de flujo

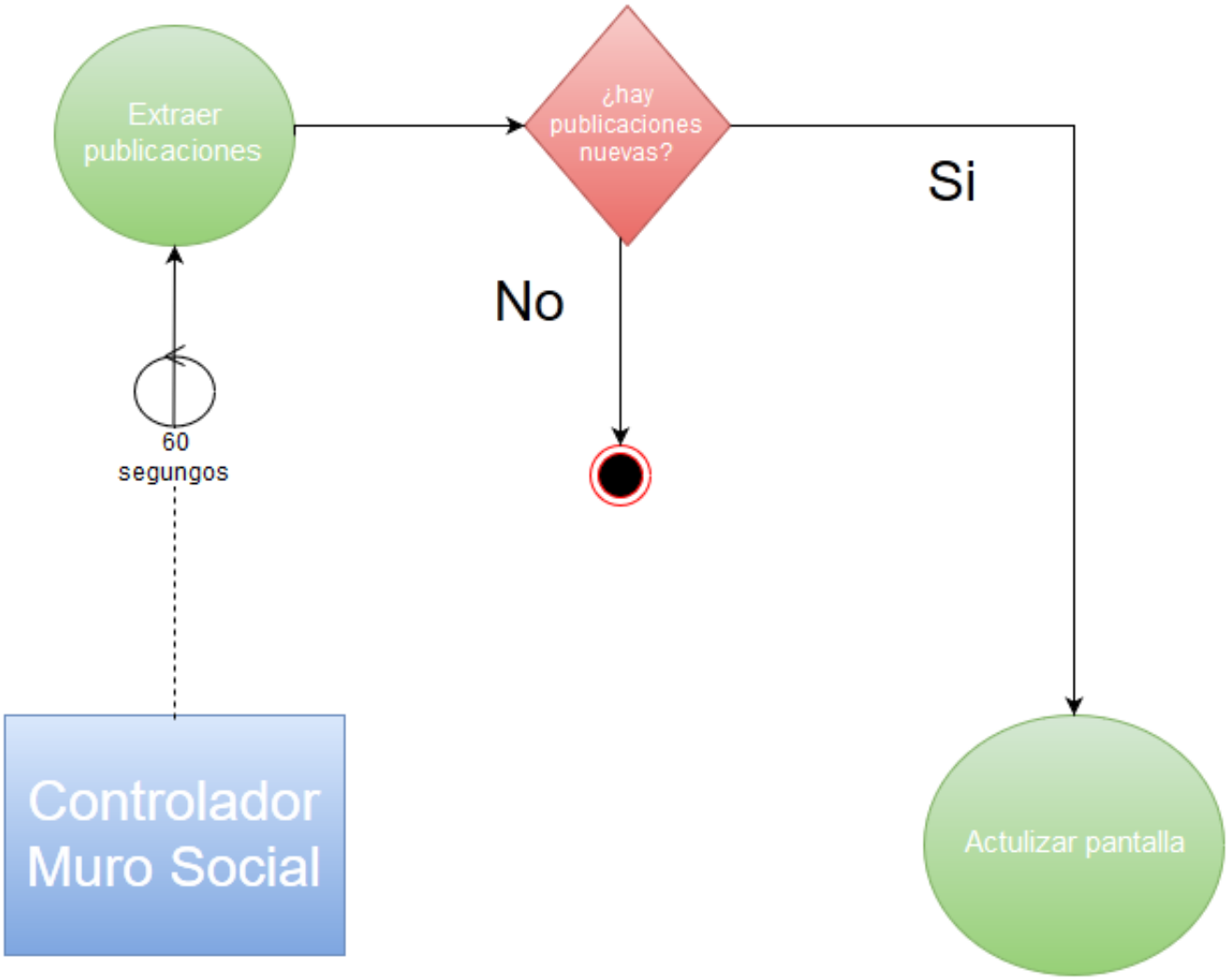


Figura 4.1: Diagrama de flujo aplicación muro social

Diagrama de clase

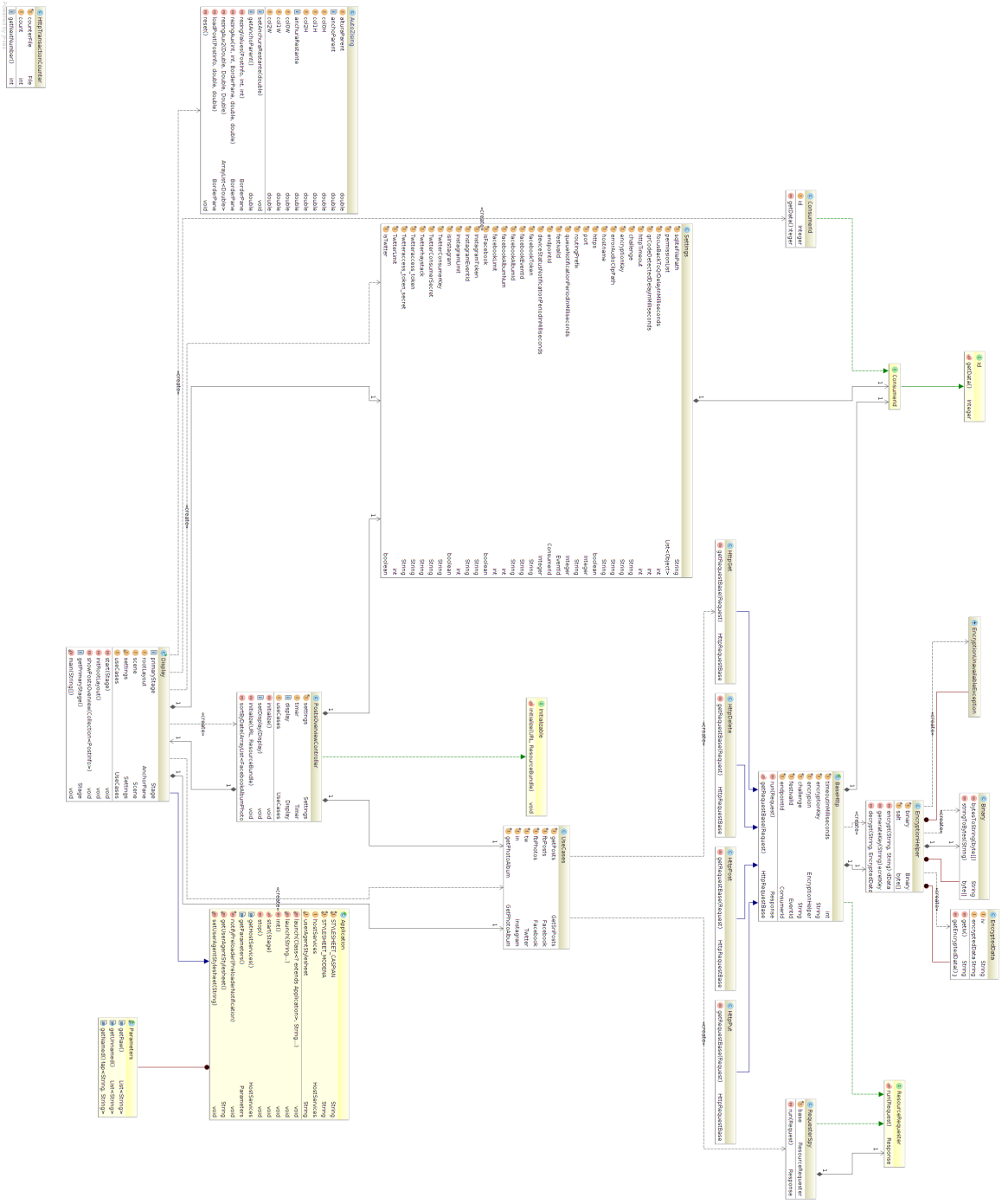


Figura 4.2: Diagrama de clases aplicación muro social

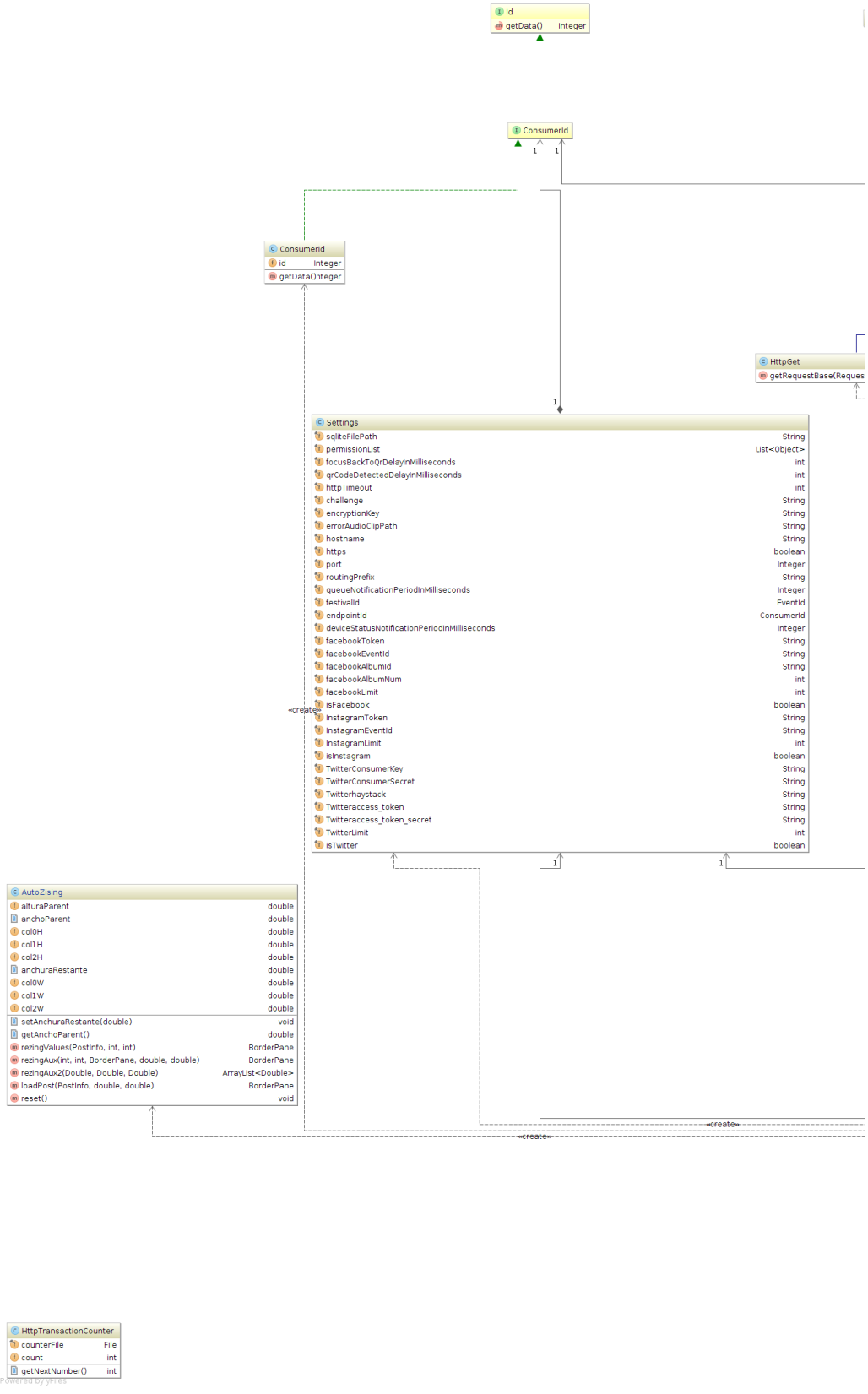


Figura 4.3: Diagrama de clases aplicación muro social ampliación 1

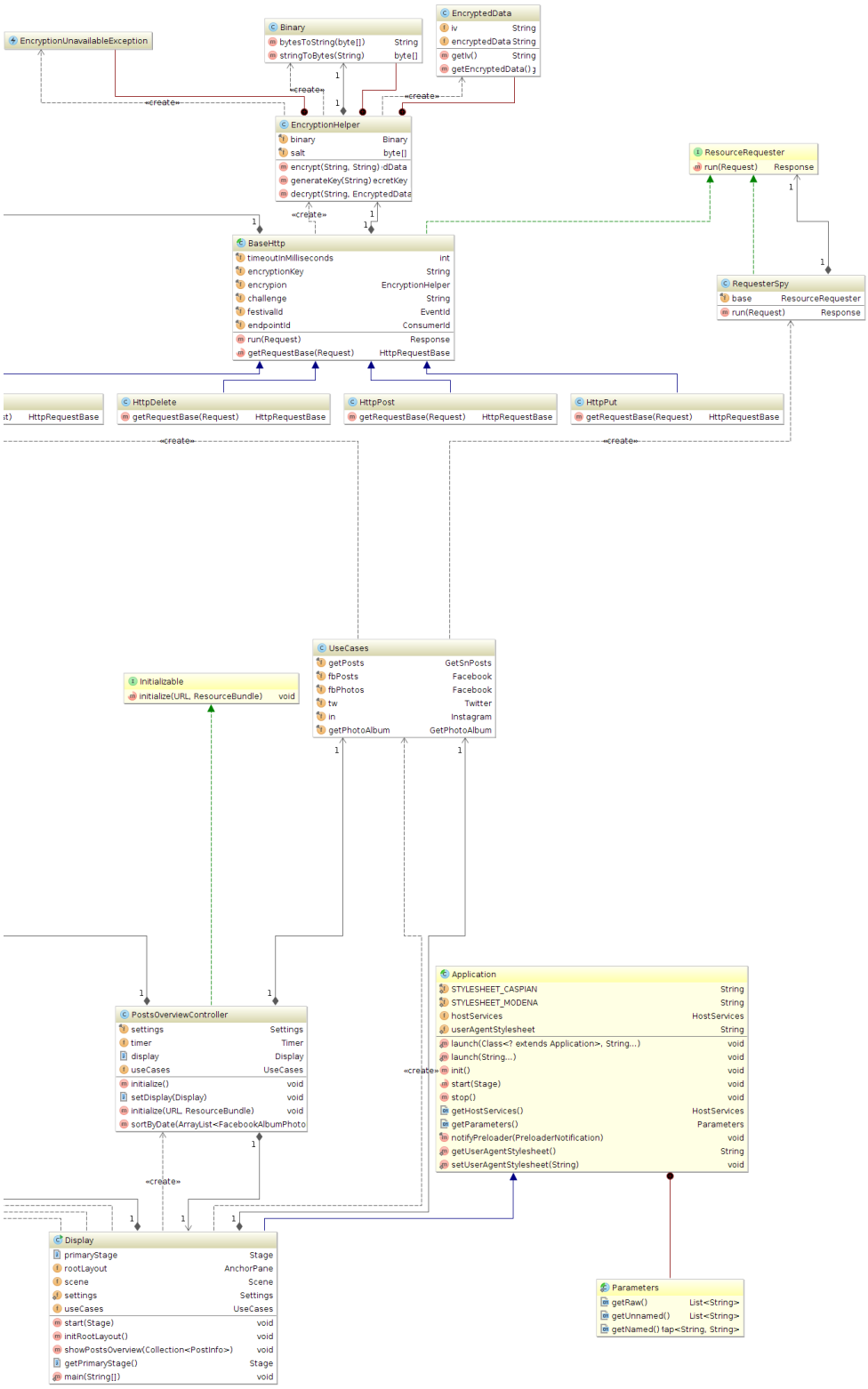


Figura 4.4: Diagrama de clases aplicación muro social aplicación 2

4.1.2. Diseño de la arquitectura del sistema

A continuación se expone el diagrama de la base de datos que corresponde a la caché local de la aplicación:

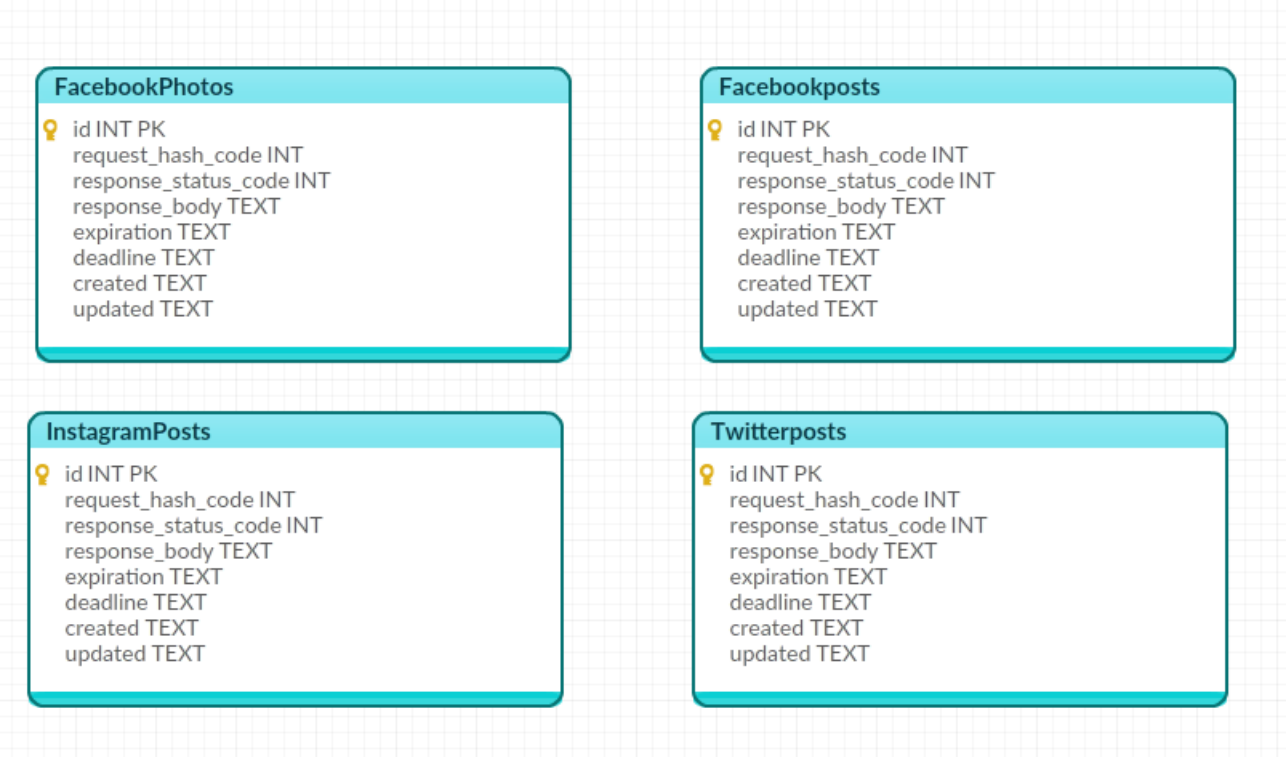


Figura 4.5: Diagrama base de datos cache local aplicación muro social

4.1.3. Diseño de la interfaz

Esta aplicación sólo tiene una pantalla que se actualiza cada minuto y muestra las publicaciones de los usuarios en las redes sociales.

La pantalla es la siguiente:

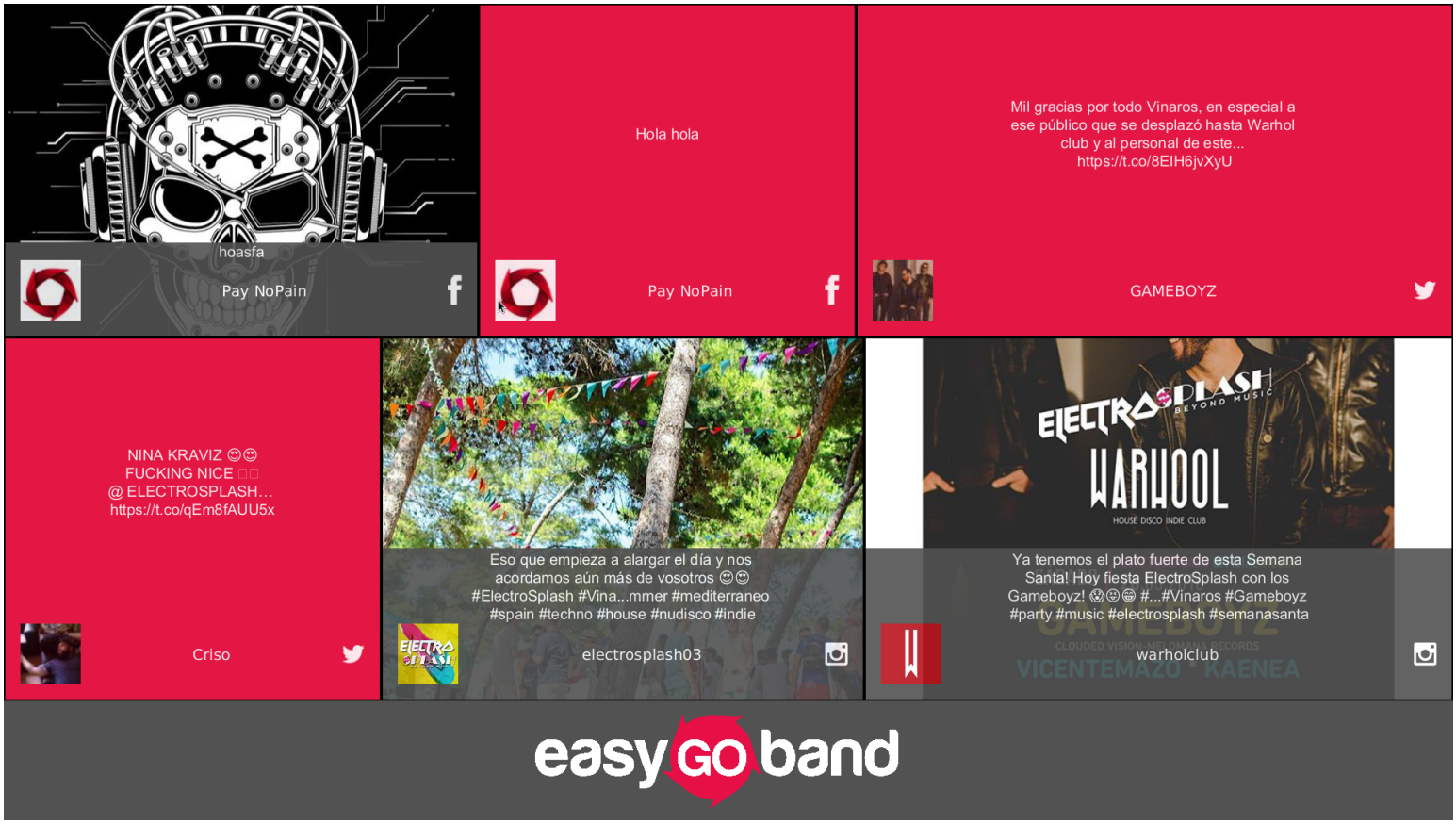


Figura 4.6: Interfaz aplicación muro social

4.2. Implementación y pruebas

4.2.1. Detalles de implementación

Esta sección del documento se estructura en cuatro subsecciones: archivo de configuración, modelo, controlador y vista.

- **Archivo de configuración:** La aplicación dispone de un archivo de configuración JSON, nombrado "settings.json" y almacenado localmente, que contiene todos los parámetros para el funcionamiento de la aplicación. Hay dos tipos de parámetros los de configuración del servidor core y los parámetros para utilizar la API de las distintas redes sociales.

  Los parámetros de configuración son los siguientes:

  - **hostname:** String que indica el hostname del servidor.
  - **port:** int que indica puerto del servidor.
  - **https:** boolean que indica si se quiere usar una conexión https.
  - **routing prefix:** string informativo.
  - **festival id:** se trata de un int que indica el id del festival.
  - **endpoint id:** se trata de un int que indica el id del endpoint en el servidor.
  - **queue notification period in milliseconds:** int que indica el tiempo máximo de espera de una notificación.
  - **challenge:** String que se utiliza para hacer el reto en la comunicación.
  - **encryption key:** se trata de un String que contiene la clave de cifrado.
  - **sqlite file path: "** String que indica dónde se creará la caché de la aplicación.
  - **pos id:** String que indica que es una petición para extraer los posts, es un parámetro informativo.
  - **pos name:** String que dará nombre a la tabla que almacenará las respuestas de las APIs.
  - **facebookToken:** String que contiene el Token de Facebook.
  - **facebookEventId:** String que contiene el id del evento de Facebook.
  - **facebookAlbumId:** String del álbum de Facebook.
  - **facebookPhotosNum:** entero que indica cuántas fotos se extraerán del álbum.
  - **facebookLimit:** se trata de un entero que indica cuántas publicaciones se extraerán del muro del evento de Facebook.
  - **isFacebook:** boolean que indica si el evento dispone de una cuenta de Facebook.
  - **InstagramToken:** String que contiene el Token de Instagram.
  - **InstagramEventId:** String que indica el id del evento en Instagram.
  - **InstagramLimit:** entero que indica cuántas publicaciones se extraerán del evento en Instagram.
  - **isInstagram:** boolean que indica si el evento dispone de una cuenta de Instagram.

- **TwitterConsumerKey:** String que contiene la clave de la aplicación de Twitter.
- **TwitterConsumerSecret:** String que contiene la clave secreta de la aplicación de Twitter.
- **Twitterhaystack:** String que contiene el haystack en Twitter.
- **Twitteraccess token:** String del Token de Twitter.
- **Twitteraccess token secret:** String del Token secreto de Twitter
- **TwitterLimit:** entero que indica cuántas publicaciones se extraerán de Twitter.
- **isTwitter:** boolean que indica si el evento dispone de una cuenta de Twitter.

- **Modelo:** La clase principal es la clase Display que extiende al objeto Application de Java. Esta clase tiene tres funciones:

  - **Cargar la configuración del programa:** carga la información almacenada en el archivo "settings.jsonütilizando la clase Settings
  - **Actualizar la interfaz:** para ello tiene un método nombrado showPostsOverview que recibe una colección de objetos post y se encarga de colocar los diferentes posts de forma dinámica por la pantalla, es decir las dimensiones del contenedor del post se calculan utilizando la clase Autozing que devuelve la altura y la anchura del contenedor de cada post utilizando un algoritmo que más adelante se explicará.
  - **Cargar la vista a partir de un archivo fxml:** previamente se ha creado un archivo fmxl que describe los diferentes elementos de la interfaz y esta clase se encarga de cargarlo para poder añadir la información de los post a estos elementos.

- **Controlador:** La clase Display tiene un controlador nombrado **PostsOverviewController** que se encarga de realizar hasta cuatro peticiones diferentes cada 60 segundos y si alguno de los post extraídos es nuevo, se actualiza la interfaz llamando al método de la clase Display showPostsOverview para actualizar la interfaz con los nuevos posts. Las peticiones que se quiere realizar se pueden configurar con los parámetros almacenados en el archivo settings.json.

  Para poder realizar esta tarea, el controlador dispone de un Timer que se ejecuta cada 60 segundos realizando la petición a las diferentes APIs y dos colecciones, la de posts actuales (.ºbservableList") y la de nuevos posts (Çollection"). Cuando se realiza una petición los post extraídos se almacenan en la colección de posts nuevos y se compara con la colección de posts actuales. Si éstas son diferentes los nuevos posts se almacenan en la colección de posts actuales y como esta colección extiende a la clases .ºbservable"se ejecuta la automáticamente el método de refresco de la pantalla con los nuevos posts.

A continuación se muestra las peticiones que se realizan para cada red social:

- **Facebook:**
  - Extraer las publicaciones del muro del evento: tbfInstagram: Petición: [1]

    ```
    https://graph.facebook.com/" "idEvento"?fields=feed.limit
    ("limite de publicaciones")%7Bfrom,message,
    id,link,full_picture,
     likes%7D&access_token="token de Facebook"
    ```

  - Extraer fotos de un álbum de Facebook:

    ```
    https://graph.facebook.com/v2.5/" "Album id"
    ?fields=photos{images,created_time,id}
    &access_token="Facebook token"
    ```

- **Twitter:** [8]
  - url:

    ```
    https://api.twitter.com/1.1/search/tweets.json?
    q="Haystack" &result_type=recent
    &count="Numero de publicaciones"
    ```

  - Cabecera:

    ```
    Auth oauth_consumer_key="ConsumerKey",
    oauth_nonce="Nonce,
    oauth_signature=Signature,
    oauth_signature_method="HMAC-SHA1,
    oauth_timestamp=Timestamp,
    oauth_token=" Access_token,
    oauth_version="1.0"
    ```

    A continuación se explican los campos de la cabecera:
    - ConsumerKey: es la clave de la aplicación de Twitter.
    - Nonce: se trata de un string aleatorio.
    - Signature: es un String que representa la firma de la aplicación y se genera con tres elementos : una firma base que se genera a partir de la url del servidor de Twitter y los parámetros de la petición, la clave secreta de la aplicación de Twitter y por último el Token secreto de la aplicación de Twitter.
    - signature method : determina el tipo de cifrado de la firma.
    - Timestamp: es un String que se genera utilizando la hora en la que se realiza la petición
    - access token: se trata del token de la aplicación de Twitter.
    - version: determina que versión de la API se va a utilizar.

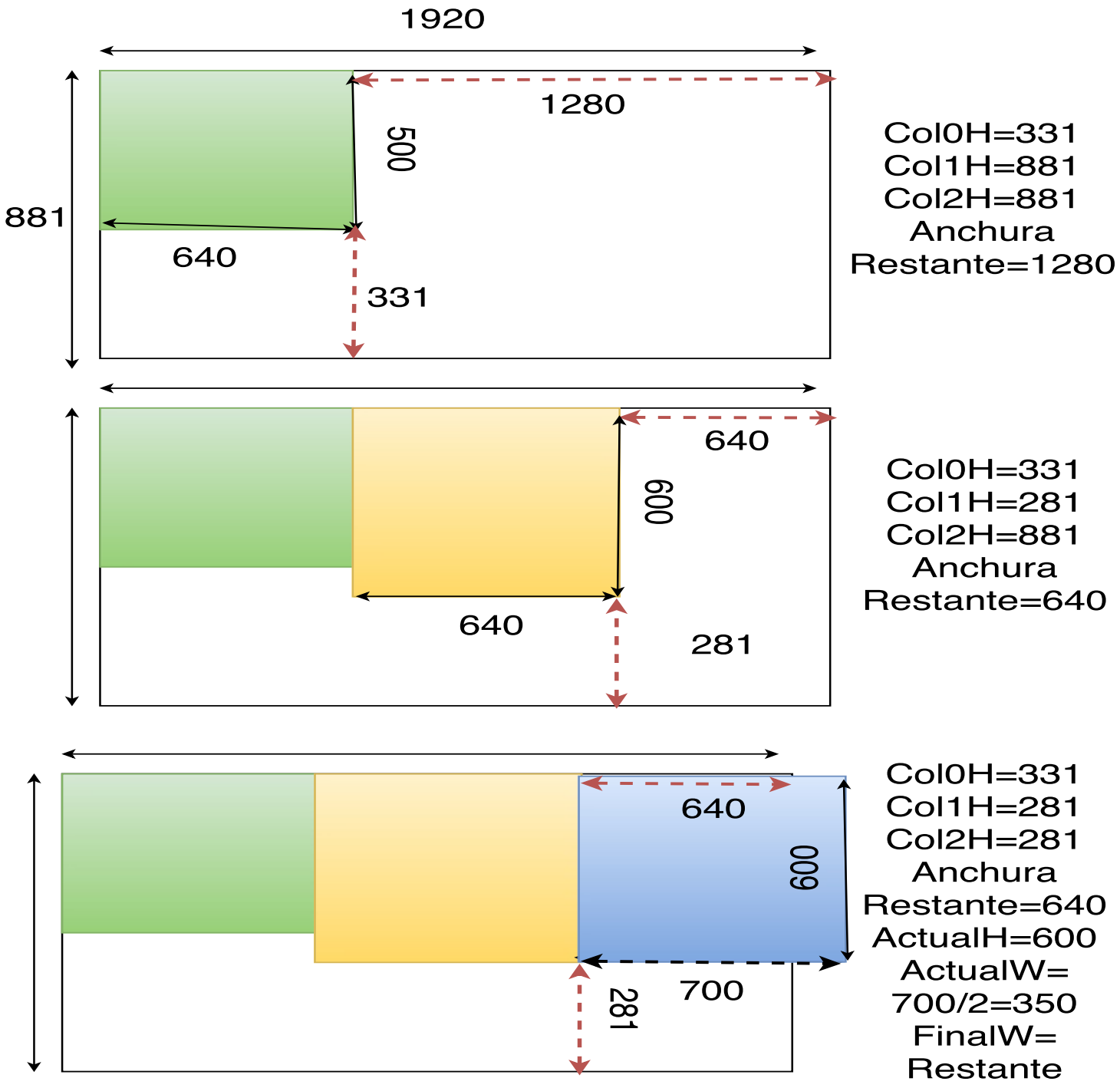
- **Instagram:** Petición: [3]

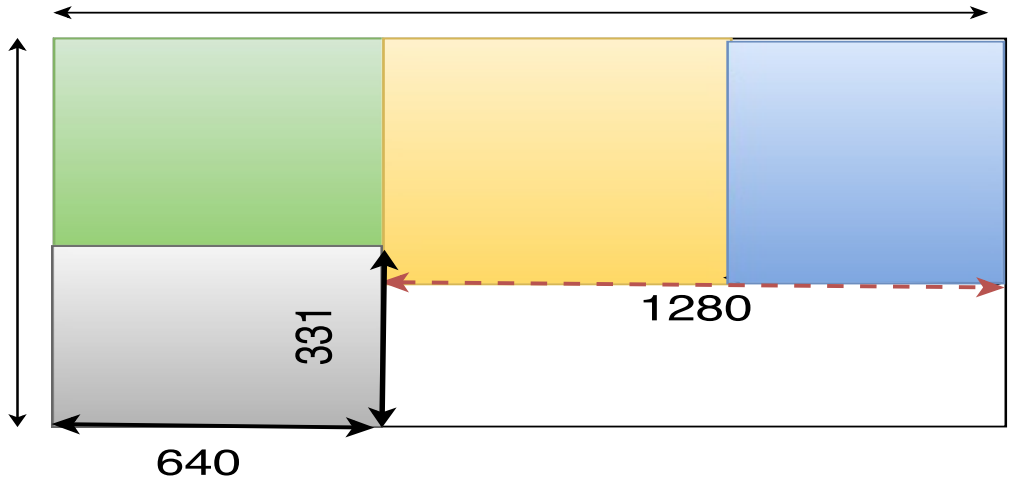
  ```
  https://api.instagram.com/v1/tags/"EventId"
  /media/recent/?access_token="Token" &count="Limit"
  ```

Finalmente tenemos la clase *Autozising* que tiene la función de calcular mediante un algoritmo la altura y la anchura del contenedor donde se va a cargar la información de cada publicación. El algoritmo que lo calcula ha sido necesario crearlo por mi parte porque el contenedor *TableView* que ofrece JavaFx no me ofrecía la apariencia deseada. El algoritmo está pensado para que en la interfaz muestre una matriz de 3 columnas y 2 filas y funciona de la siguiente manera:

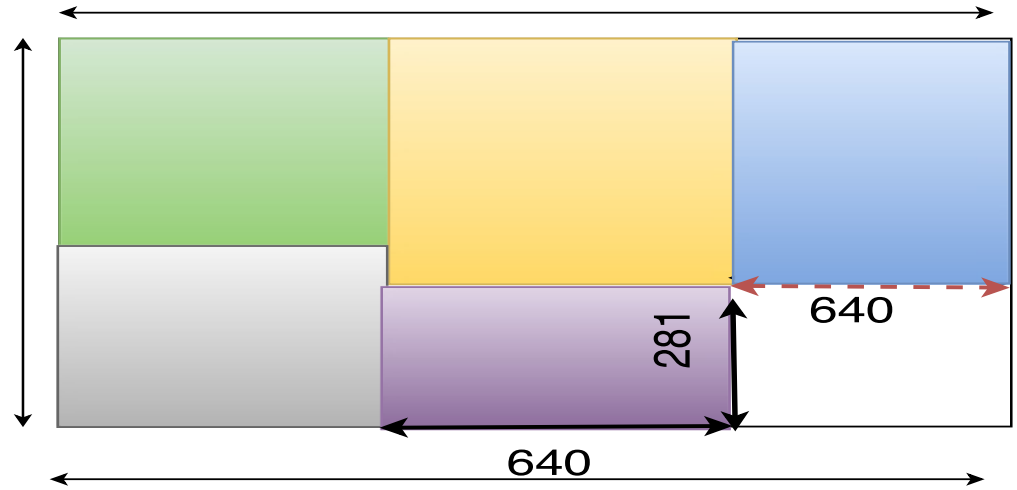
- Primero hay que distinguir que hay dos tipos de publicaciones con foto o sin foto.
- Después hay qué tener en cuenta en qué fila y en qué columna se va a colocar, ya que las publicaciones se irán añadiendo de izquierda a derecha y de arriba a abajo.
- Por último si alguna publicación tiene unas dimensiones que superan las del rectángulo se le asigna un tamaño igual a la mitad de la altura o/y la anchura restante dependiendo por dónde sobresalga. En el caso de que queden huecos libres, se le sumará el tamaño restante para que ocupen toda la superficie (por ejemplo en el caso que se encuentre en la última columna y sobresalga por anchura y se le asigne la mitad del tamaño y ese tamaño no sea suficiente para cubrir la anchura restante).

Seguidamente se muestra una representación gráfica del algoritmo:

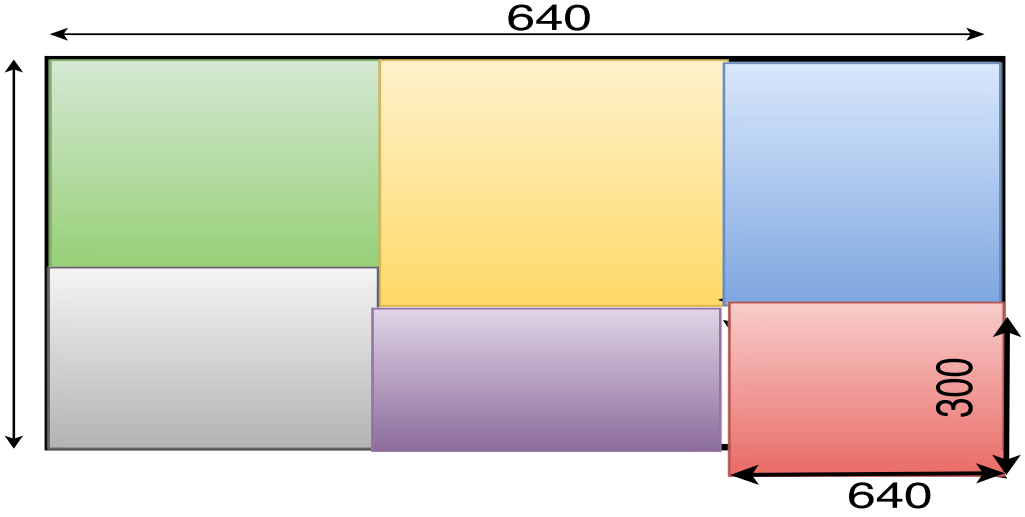

1920
1280
500
881
640
331
Col0H=331
Col1H=881
Col2H=881
Anchura
Restante=1280
640
600
640
281
Col0H=331
Col1H=281
Col2H=881
Anchura
Restante=640
640
600
700
281
Col0H=331
Col1H=281
Col2H=281
Anchura
Restante=640
ActualH=600
ActualW=
700/2=350
FinalW=
Restante



Col0H=0
Col1H=281
Col2H=281
Anchura
Restante=1280



Col0H=0
Col1H=0
Col2H=281
Anchura
Restante=640



Col0H=0
Col1H=0
Col2H=281
Anchura
Restante=640
AlturaFinal=
Col2H=281

- **Vista:** A continuación se muestran dos diagramas que describen los elementos que forman la interfaz de la aplicación.

El primer diagrama corresponde a la escena principal de la interfaz que está formada por un BorderPane que hace la función de contenedor, en el cuál se insertan dinámicamente las publicaciones que se organizarán en el FlowPane las cuales se encuentran en el segundo diagrama que representa la estructura de las publicaciones en la interfaz del muro social.

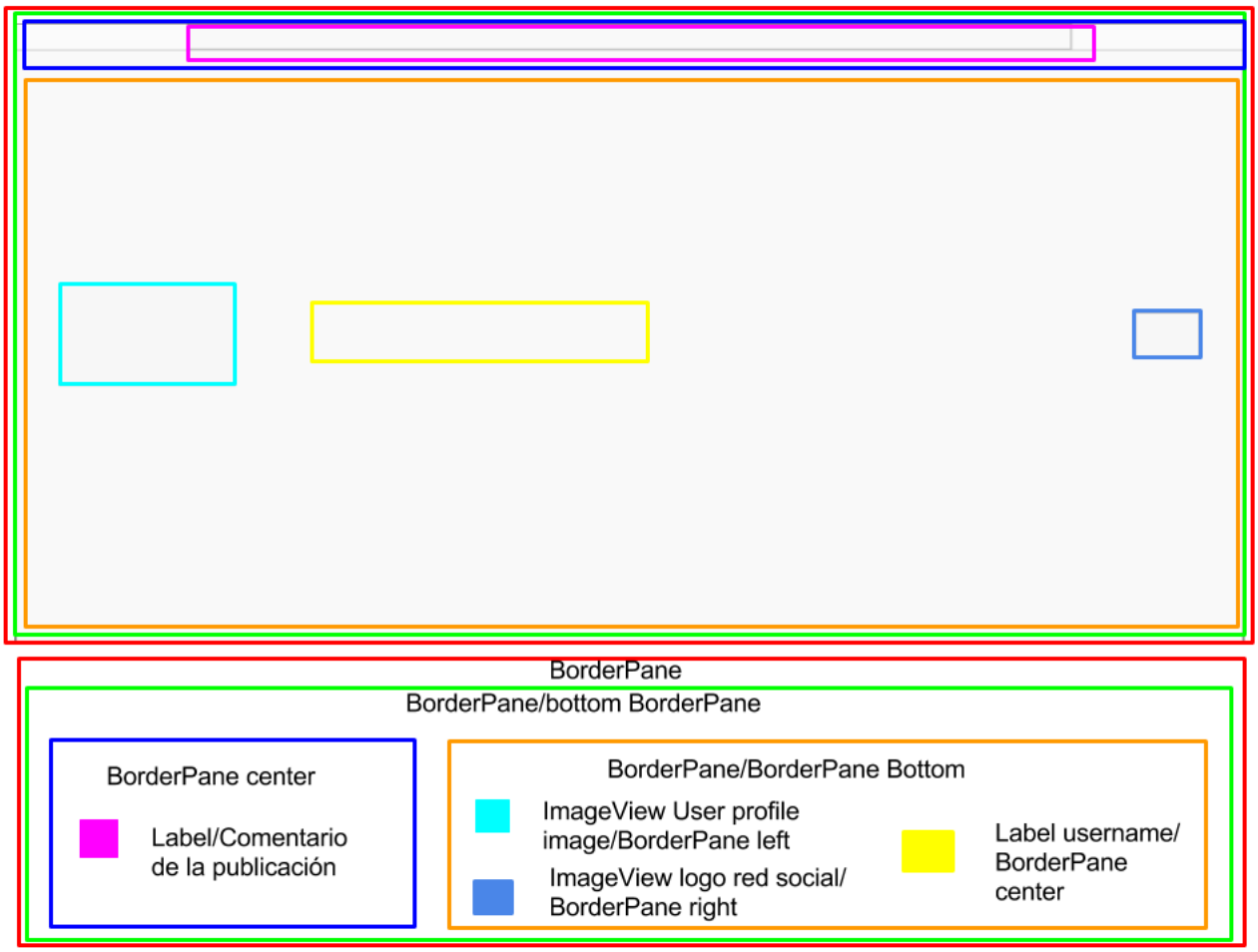


Figura 4.7: Estructura del contenedor de las publicaciones

Figura 4.8: Estructura de la interfaz de la aplicación muro social

4.2.2. Verificación y validación

Como se ha comentado anteriormente en el documento, la empresa utiliza la metodología TDD a la hora de desarrollar nuevo software. Por lo tanto, para toda la parte que forma el Modelo de la aplicación se han creado los test pertinentes antes de realizar la implementación y se han ejecutado para comprobar que todas las peticiones se envían y reciben correctamente.

Por otra parte se realizó un prueba de rendimiento que consistió en dejar un periodo considerable de tiempo (una semana) la aplicación ejecutándose para observar su comportamiento a la hora de actualizar la interfaz cuando se producen un gran número de publicaciones en las redes sociales y encontrar errores en la ejecución. El resultado de está prueba fue satisfactorio y la aplicación se comportó de la forma esperada cuando se produjo un gran flujo de publicaciones en las redes sociales.

Capítulo 5

Aplicación totem

5.1. Análisis y diseño del sistema

5.1.1. Análisis del sistema

Requisitos

- Realizar una aplicación con tres funcionalidades: consultar el saldo, actualizar la pulsera y el *photocall*.
- La interfaz tiene que adaptarse al estilo del festival de música Madcool.
- La resolución de las interfaces de la aplicación tiene que ser de 1024p768.
- Las fotos tomadas con la aplicación no se deben almacenar en local y no tiene que ser de un tamaño mayor a 1MB.
- La aplicación tiene que optimizarse para que se ejecute con normalidad en la Raspberry pi 3.
- El diseño de las interfaces debe de ser muy intuitivo y fácil de usar.

Diagrama de flujo

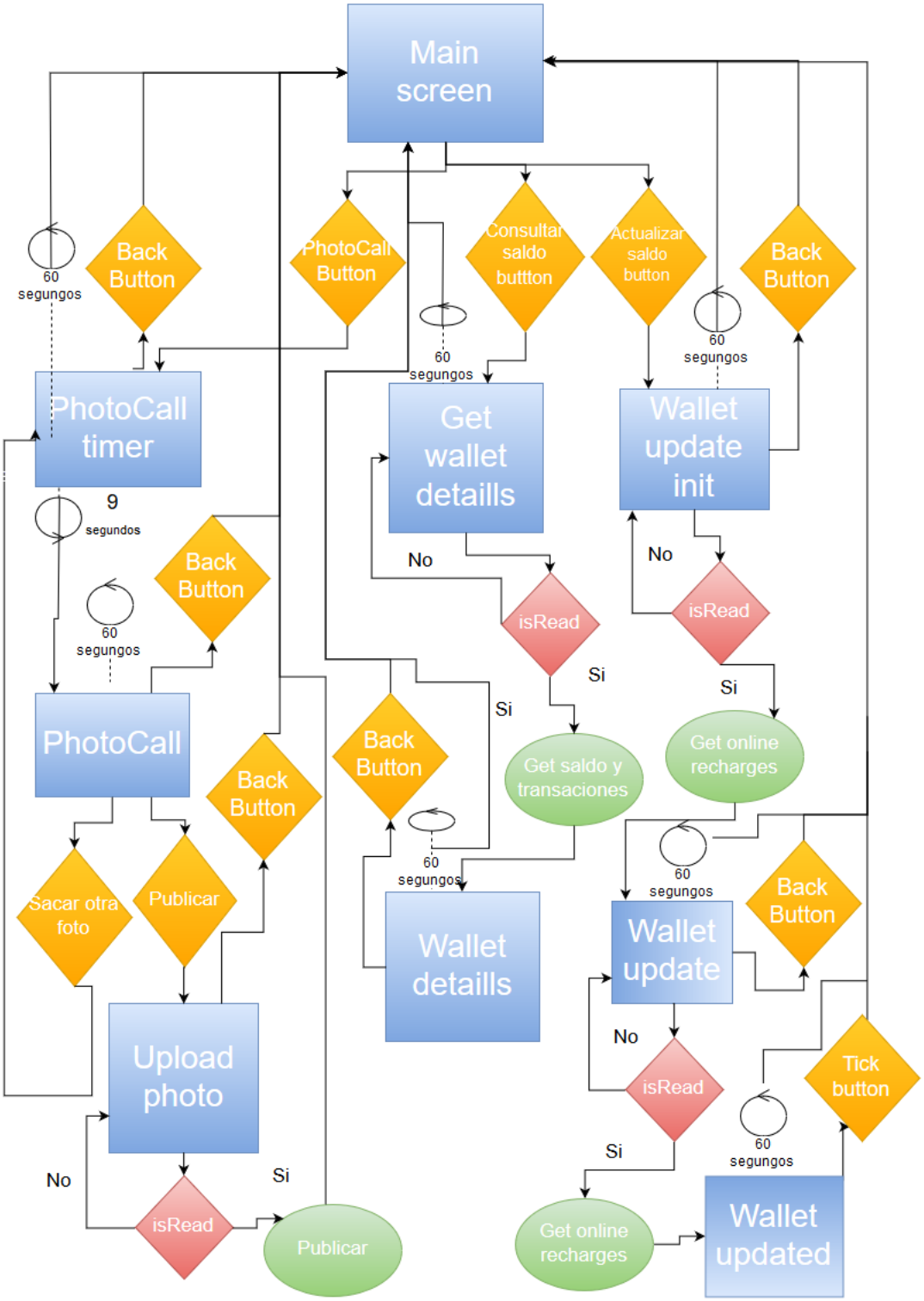


Figura 5.1: Diagrama de clases aplicación Tótem

Diagrama de clases

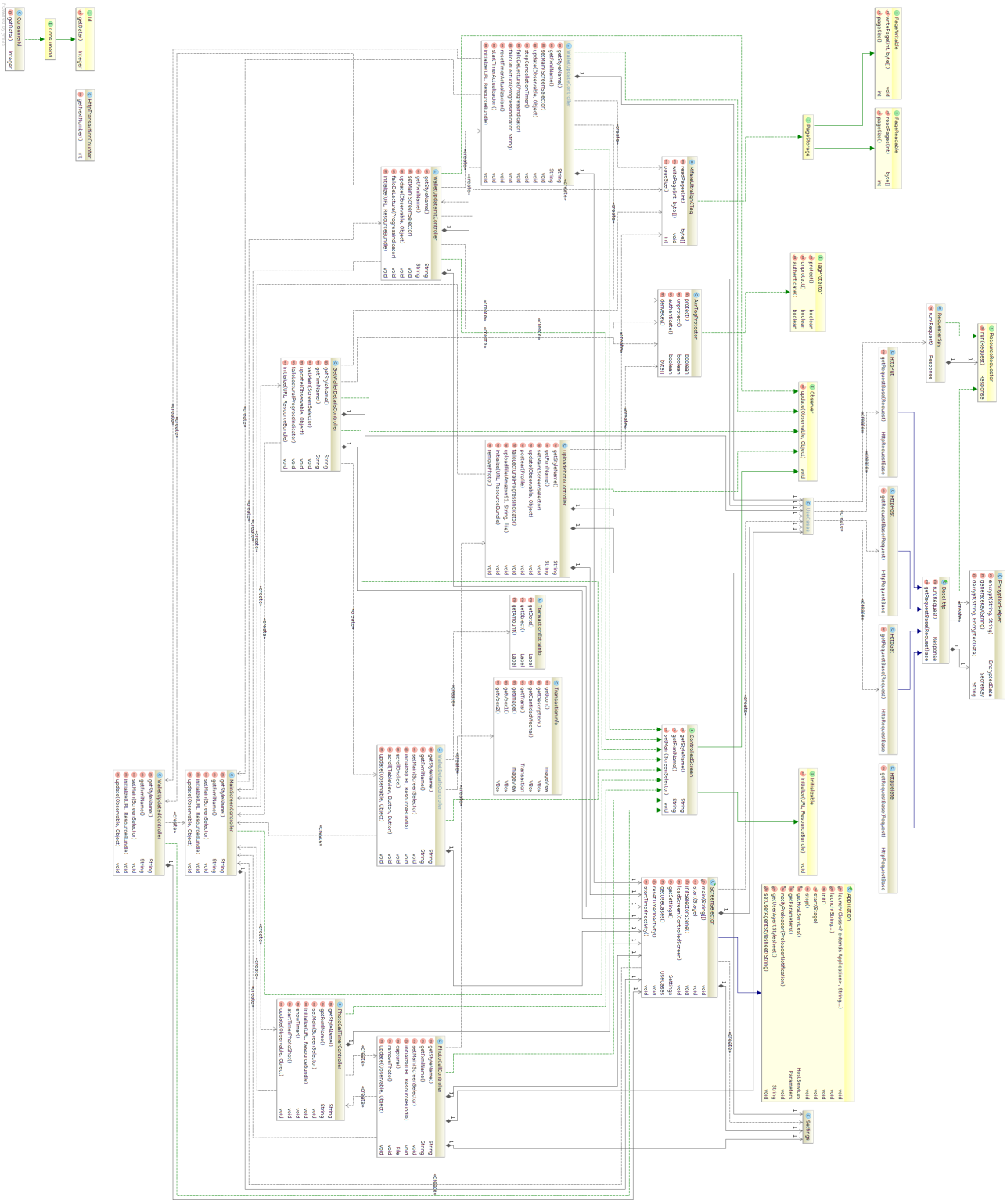


Figura 5.2: Diagrama de clases aplicación Tótem

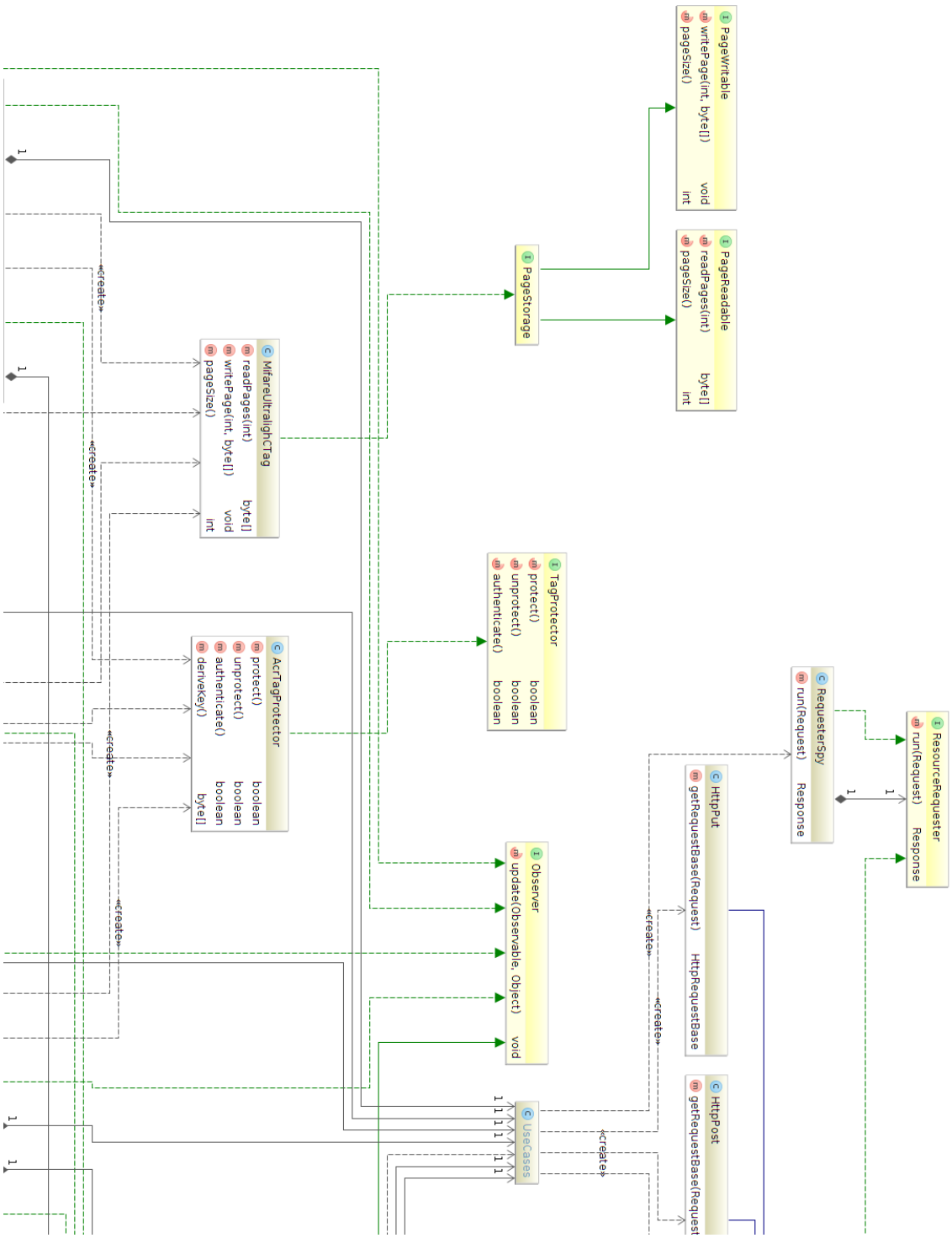


Figura 5.3: Diagrama de clases aplicación Tótem ampliado 1

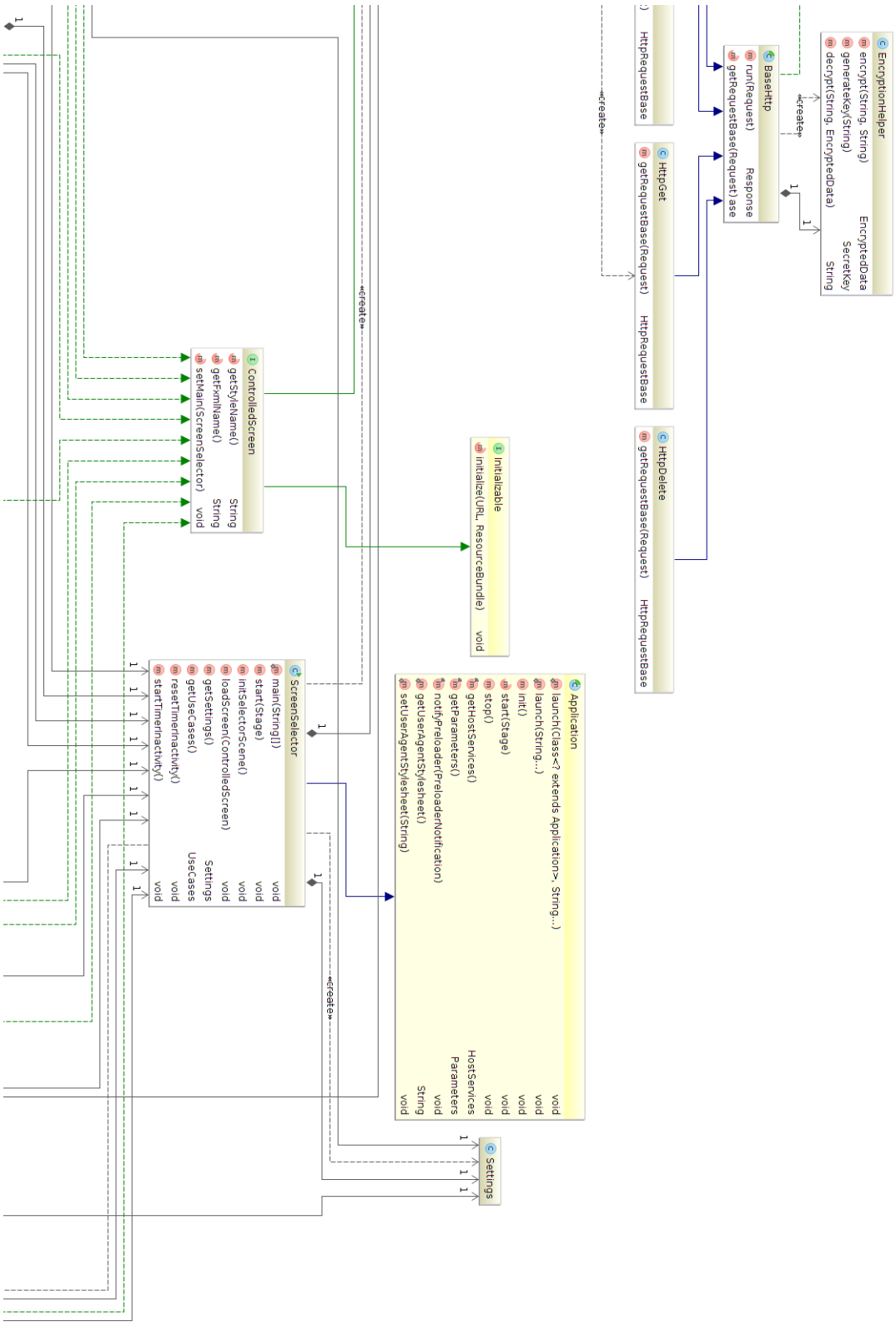


Figura 5.4: Diagrama de clases aplicación Tótem ampliado 2

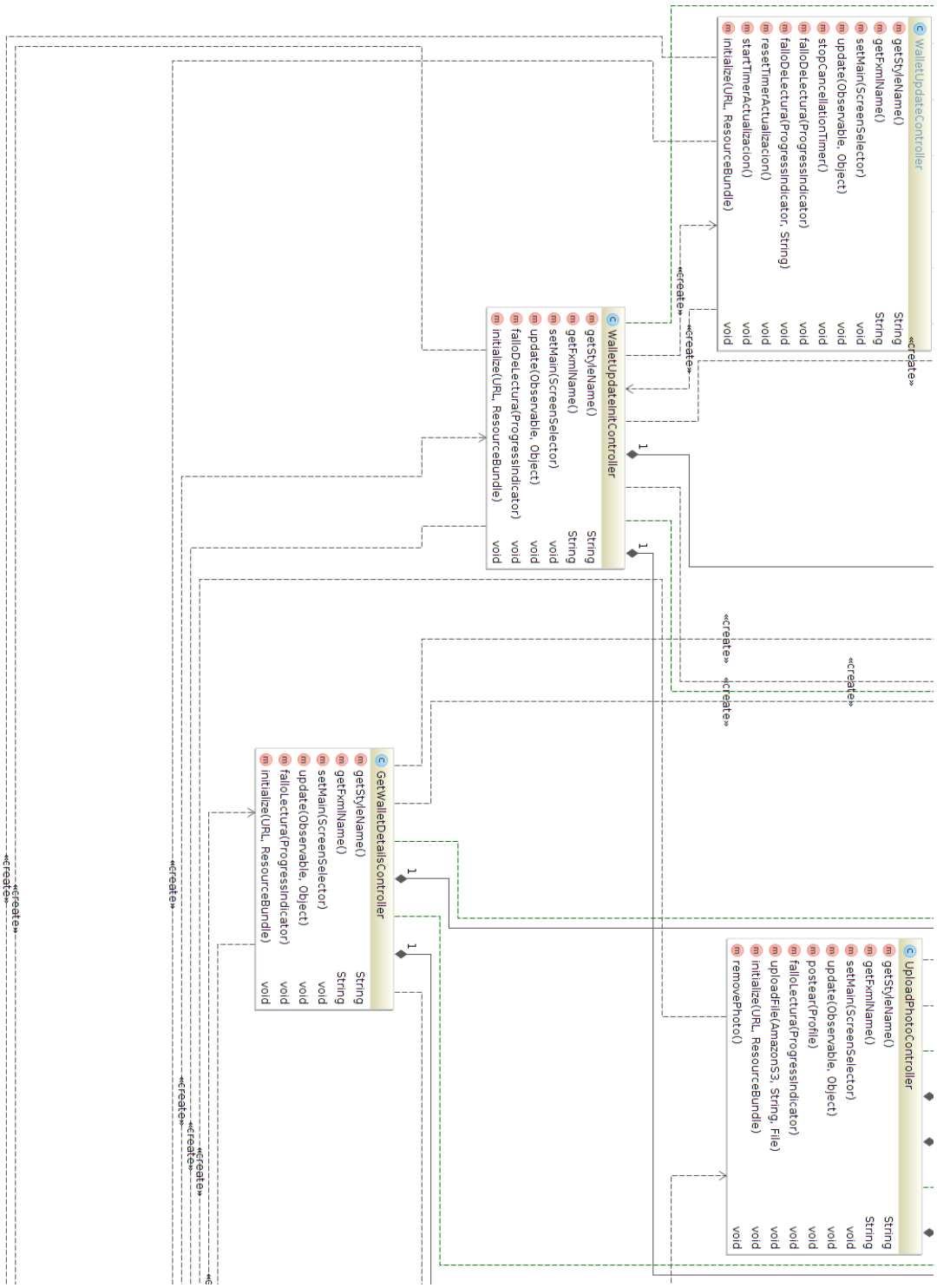


Figura 5.5: Diagrama de clases aplicación Tótem ampliado 3

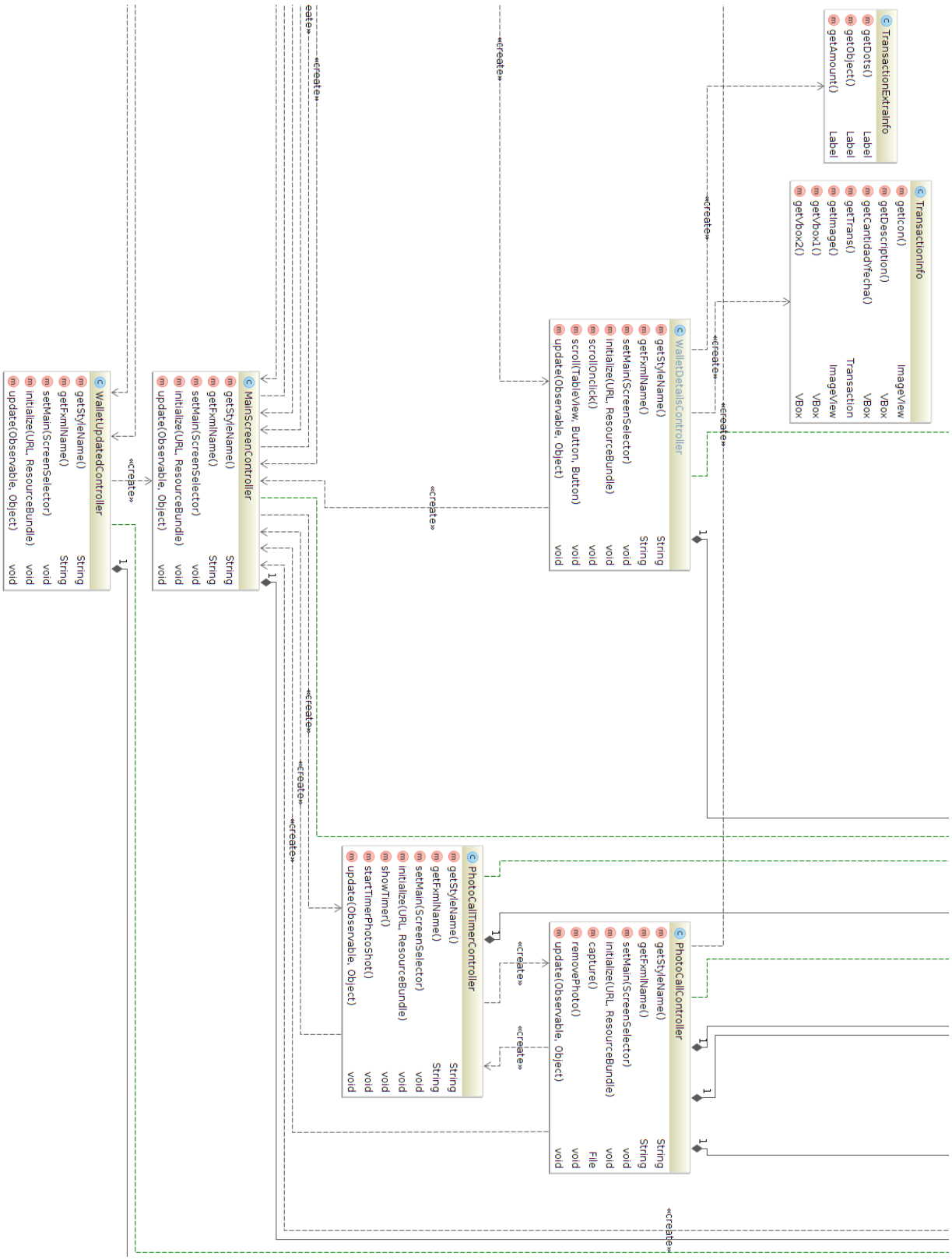


Figura 5.6: Diagrama de clases aplicación Tótem ampliado 4

5.1.2. Diseño de la arquitectura del sistema

A continuación se expone el diagrama de la base de datos que corresponde a la cache local de la aplicación:

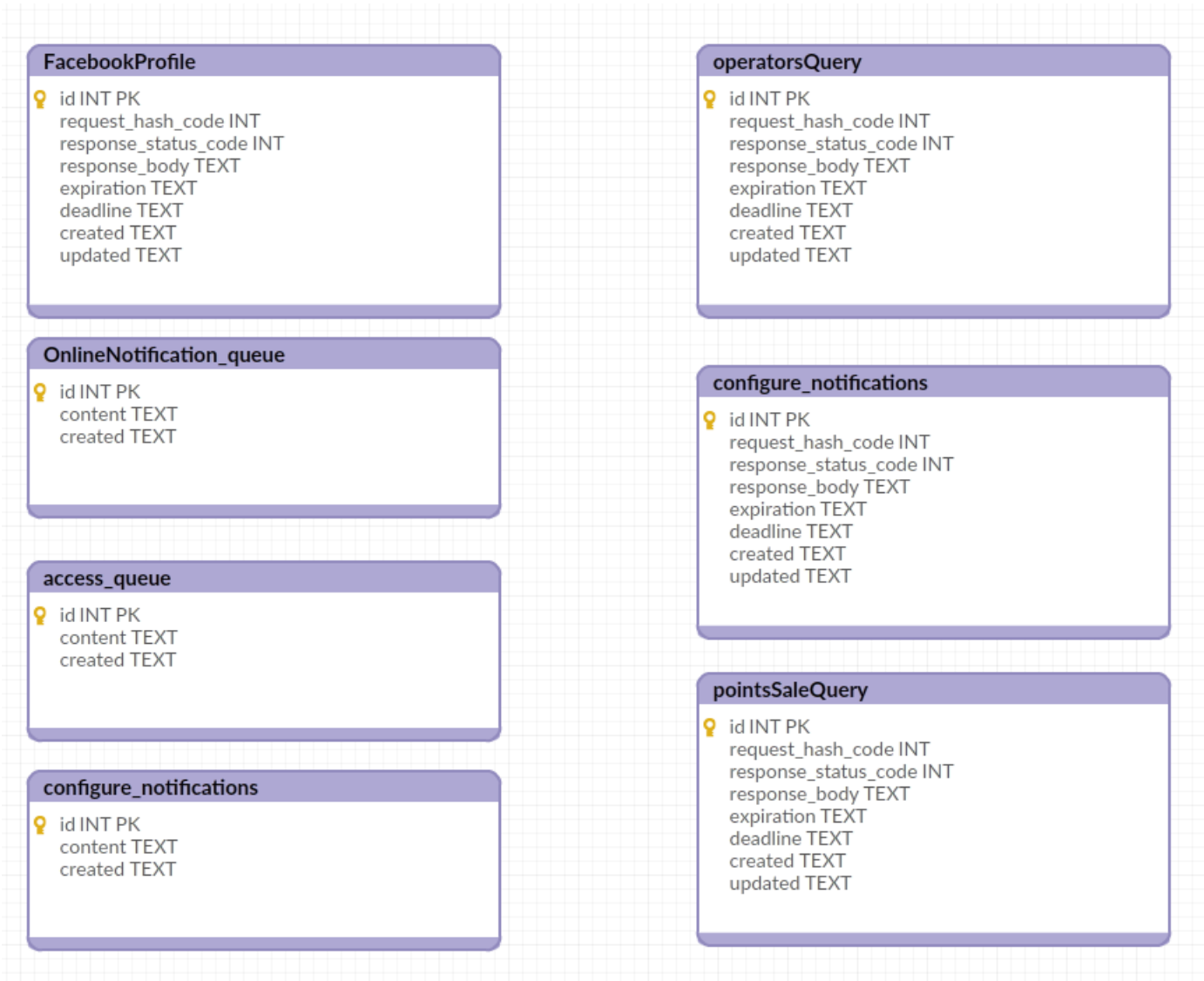


Figura 5.7: Diagrama base de datos de la cache local de la aplicación del Tótem

5.1.3. Diseño de la interfaz

Esta aplicación consta de 6 pantallas diferentes en las que se puede navegar utilizando los botones y el lector de la pulsera NFC.

Las pantallas son las siguientes:

- **Pantalla 1/Pantalla principal:** Esta es la pantalla principal donde se puede seleccionar la funcionalidad de la aplicación a utilizar.

- **Pantalla 2/ Pantalla Cuenta atrás:** Esta pantalla se carga cuando se presiona el botón de sacar foto y es una cuenta atrás para que los usuarios se dispongan a fotografiarse.

- **Pantalla 3/ Pantalla Pulsera:** Pantalla intermedia que aparece cuando es necesario leer la información almacenada en la pulsera.

- **Pantalla 4/ Pantalla actualizar pulsera:** Pantalla para actualizar el saldo de la pulsera. Como indica el nombre, su función es actualizar la pulsera cuando se hace una recarga.

- **Pantalla 5/ Pantalla consulta saldo :** Esta pantalla es la de consulta del saldo donde el usuario puede consultar su saldo y las transacciones realizadas.

- **Pantalla 6/ Pantalla *photocall*:** Por último, esta es la pantalla del PhotoCall, donde el usuario puede ver la fotografía tomada y decidir si la publica en Facebook o quiere sacarse otra fotografía.

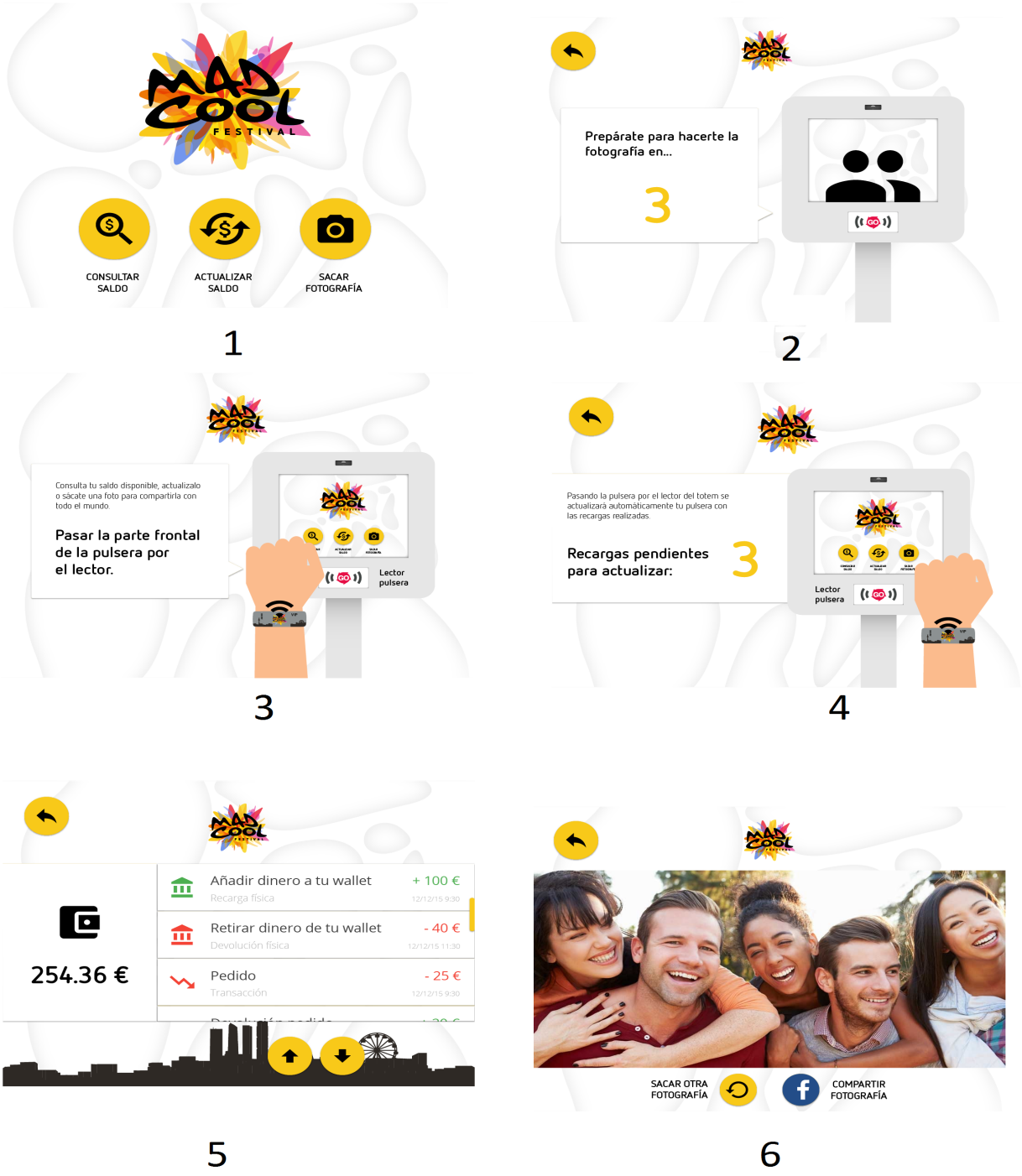


Figura 5.8: Pantallas aplicación Tótem

5.2. Configuración de la pulsera NFC

En esta sección se explica el proceso de configuración de la pulsera NFC y qué información contiene ésta. En realidad, en la implementación sólo se han modificado los módulos que ya existían antes de empezar este proyecto para añadir la información del Token y el userId de Facebook, pero es interesante saber cómo funciona este proceso para entrar en contexto.

El proceso de configuración de la pulsera es el siguiente:

- **Escribir la información de identificación en la pulsera:** La pulsera contiene una etiqueta que la identifica, con la cual se realizarán todas las peticiones al servidor (Core de la aplicación). Esta información es escrita en la pulsera por un operario utilizando un dispositivo llamado Famoco(Professional secure Android NFC reader) durante el proceso de activación (este proceso es opaco para los usuarios). Para evitar problemas de seguridad las pulseras están cifradas utilizando AES3.
- **Vinculación de la pulsera con la cuenta del usuario:** Ahora ya en mano del usuario, éste, utilizando la aplicación móvil de MadCool, podrá vincular su cuenta de easyGoBand con su pulsera introduciendo el número de identificación, que está escrito en la parte frontal de la pulsera, y en el formulario de vinculación de la aplicación móvil. Con esto se consigue que cuando el usuario pase su pulsera por el lector del tótem éste lea el mismo número que el usuario introdujo, el cual está almacenado en la pulsera NFC y podrá extraer la información solicitada utilizado el identificador (WalletId) en la peticiones al servidor del Core.

5.3. Implementación y pruebas

5.3.1. Detalles de implementación

Esta aplicación es más compleja que la anterior ya que es una aplicación multi-pantalla e interactiva. Por ello se va a dividir esta sección en cuatro secciones: Archivo de configuración, modelo, controlador y vista.

- **Archivo de configuración :** La aplicación dispone de un archivo de configuración JSON, nombrado settings.json y almacenado en local que contiene todos los parámetros para el funcionamiento de la aplicación. Hay dos tipo de parámetros, los de configuración del servidor core y los parámetros para utilizar la API de las distintas redes sociales. Los parámetros de configuración son los siguientes:
  - **hostname:** *String* que indica el hostname del servidor.
  - **port:** int que indica puerto del servidor.
  - **https:** boolean que indica si se quiere usar una conexión https.
  - **routing prefix:** string informativo.

- **festival id:** se trata de un int que indica el id del festival.
- **endpoint id:** se trata de un int que indica el id del endpoint en el servidor.
- **queue notification period in milliseconds:** int que indica el tiempo máximo de espera de una notificación.
- **challenge:** *String* que se utiliza para hacer el reto en la comunicación.
- **encryption key:** se trata de un *String* que contiene la clave de cifrado.
- **sqlite file path:** es un *string* que indica donde se creará la caché del la aplicación.
- **pos id:** *String* que indica que es una petición para extraer los posts, es un parámetro informativo.
- **pos name:** *String* que dará nombre a la tabla que almacenará las respuestas de las APIs.
- **bucketId:** *String* que contiene el id del *Bucket de Amazon S3.*.
- **bucketKey:** *String* que contiene la clave para acceder al *Bucket de Amazon S3.*
- **bucketName:** *String* que contiene el nombre que identifica el *Bucket de Amazon S3.*
- **sync interval in milliseconds:** entero que determina el intervalo de sincronización con el *Bucket de Amazon S3.*

- **Modelo:** Como anteriormente se ha comentado esta aplicación tiene tres funcionalidades, consultar el saldo de tu pulsera, actualizar la pulsera, y utilizar el *photoCall* para sacarse un foto y publicarla en Facebook. Sabiendo esto la aplicación tiene los siguientes casos de uso:

  - **Consultar saldo:**
    - *getBalanceFromTag:* este caso de uso recibe un *tag* que es leído de la pulsera del usuario previamente y se lee el parámetro *amount* del *tag* que indica el saldo disponible en la pulsera.
    - *getTransactions:* igual que el caso de uso anterior, utilizando el *tag* de la pulsera se realiza una petición al servidor para extraer la lista de transacciones que ha realizado el usuario utilizando el *walletId* que está almacenado como un parámetro del *tag*.
  - **Actualizar saldo:**
    - *getOnlineRecharges:* este caso de uso realiza un petición al servidor utilizando el *walletId* para obtener una lista de recargas online que no están actualizadas en ese momento con los datos de la pulsera e indica en la base de datos del servidor que los elementos de esta lista han sido actualizados.
    - *cancelOnlineRechargeWriting:* este caso de uso sirve para marcar que las recargas obtenidas con el caso de uso anterior no se han actualizado correctamente en la pulsera y las marca en el servidor como pendientes, para que se pueda volver a repetir el proceso de actualización de la pulsera.
    - *updateTagWithOnlineRecharges:* este caso de uso se encarga de actualizar la pulsera con los nuevos datos obtenidos del caso *getOnlineRecharges*.

- **PhotoCall:**
  - *postPhoto:* este caso de uso se encarga de publicar la fotografía tomada en el muro de Facebook del usuario, utilizando su pulsera para sacar la fotografía. Para realizar esta acción son necesarias tres cosas:
    - *La url de la fotografía a publicar:* como necesitamos la url de la fotografía antes de realizar la petición se sube a un bucket de Amazon y se utiliza la url asociada a esa fotografía para publicarla en Facebook.

    - *Facebook userId:* es necesario el id del usuario de Facebook para poder indicar en la petición en qué lugar publicar la fotografía. Para obtener el id que estará almacenado en la pulsera, el usuario cuando decida vincular su pulsera con su cuenta *easyGoband* con la aplicación móvil tendrá que acceder con su cuenta de Facebook y en la respuesta del servidor de Facebook viene el id del usuario de Facebook.

    - *Token con permisos de publicación:* [2] para poder publicar como si fueras el usuario la aplicación necesita un underlineToken de usuario, con permisos de publicación, para poder obtener este *token* se necesita el consentimiento del usuario. Entonces cuando el usuario haya accedido a la aplicación móvil con su cuenta de Facebook le aparecerá un cuadro de diálogo preguntándole si permite a la aplicación publicar en su nombre, si acepta se obtendrá un *token* con los permisos de publicación. Pero este *token* es de corta duración, es decir, que tiene un tiempo de vida de 2 horas. Para evitar que el usuario tenga que está pendiente de los permisos, se realizará un petición al servidor de Facebook para obtener un *token* de larga duración a partir del *token* obtenido. La petición es la siguiente : [1]

      ```
      https://graph.facebook.com//oauth/access_token
      ?grant_type=fb_exchange_token&client_id="Facebook client id"
      &client_secret="Facebook app secret"
      &fb_exchange_token="Token de corta duración"
      ```

    Ahora que ya tenemos lo que necesitamos para realizar la publicación de la fotografía en el muro del usuario solo hay que realizar la petición, que es la siguiente:

    ```
    https://graph.facebook.com/v2.5/"Facebook user id"
    /photos?url="url de la fotografía"
    &access_token="Token con permisos de publicación"
    ```

- **Controlador:** La aplicación tiene dos tipos de controladores, el controlador de escenas que se encarga de gestionar la navegación entre las escenas de la aplicación y el controlador de cada escena que se encarga de gestionar las funcionalidades de éstas.

  - **Controlador de escenas:** El controlador de escenas tiene una única funcionalidad:
    - *loadScene:* este método se encarga de cargar la escena del controlador que se le pasa como parámetro utilizando las direcciones donde se encuentran el archivo fxml y el archivo css de la escena que se quiere cargar. Los cuales se pueden extraer de cada objeto controlador utilizando los métodos getStyleName y getFxmlName.
  - **Controladores de cada escena:** Todos los controladores de las escenas tienen una cosa en común: se encargan de cargar cada elemento dinámico de la escena, por ejemplo imágenes, botones personalizados, etc. Por otro lado, todos los controladores menos el de la pantalla principal administran la acción de pulsar el botón de volver a la pantalla anterior y tienen un *timer* que si no se produce ninguna interacción con la aplicación pasados 60 segundos carga automáticamente la pantalla principal y eliminan la información que la escena había almacenado. Esto se ha añadido para controlar que nadie se deje su información a la vista innecesariamente. A continuación se expondrán las funcionalidades particulares de cada escena.
    - **Controlador pantalla principal:** Este controlador tiene la función de capturar los eventos de los botones que lo forman y llamar al controlador de escenas para que haga la transición a la pantalla asignada a cada botón.

    - **Controlador getWalletDetaills:** este controlador tiene la función de leer la información que contiene la pulsera NFC: información sobre el saldo y las transacciones realizadas.
      Para realizar esta tarea este controlador implementa la interfaz *Observer* para capturar el evento de lectura de la pulsera y utiliza la librería *MifareUltralightC* para leer la información recogida en la lectura. En caso de que se recoja un error de lectura de la pulsera se muestra un mensaje para avisar al usuario que tiene que pasar otra vez la pulsera por el lector
    - **Controlador walletDetaills:** este controlador se encarga de mostrar la información obtenida de la escena getWalletDetaills y de gestionar los siguientes eventos capturados:
      - *Botones de desplazamiento vertical:* este administrador de eventos se encarga de gestionar los botones de desplazamiento por la tabla de transacciones, para ir desplazándose de arriba a abajo por la tabla. Hay que apuntar que este administrador se encarga tanto de gestionar la tabla de transacciones como la tabla de información extra de la transacción, haciendo la misma tarea.
      - *MouseClick en la tabla de transacciones:* este administrador de eventos se encarga de detectar cuando alguna de las filas de la tabla transacciones se ha pulsado. Si se detecta una pulsación la vista de la tabla se traslada al elemento pulsado y solo si es una transacción que contenga un carro de compra, es decir que contenga artículos comprados o devueltos, se abrirá una tabla emergente (tabla de información extra) que contendrá la información más detallada sobra la transacción.

- **Controlador walletUdateInitController:** este controlador al igual que el controlador walletDetaills implementa la interfaz *Observer* para capturar el evento de lectura de la pulsera y utiliza la librería *MifareUltralightC* para leer la información de la pulsera. En este caso este controlador utiliza el tag extraído durante la lectura para obtener el número de recargas online pendientes, utilizando el método getOnlineRecharges. En el caso de que haya alguna recarga pendiente este controlador carga la escena walletUpdate, en caso contrario se carga la escena walletUpdated.
- **Controlador walletUpdateController:** este controlador también implementa la interfaz Observer para leer la pulsera. Con el tag y las transacciones obtenidas en la escena walletUdateInitController, se realiza la actualización de la pulsera utilizando el método updateTagWithOnlineRecharges. Seguidamente se carga la escena walletUpdated.
- **Controlador walletUpdatedController:** este controlador sólo tiene que gestionar el evento de pulsación del botón tick para que cuando se pulse se cargue la pantalla principal
- **Controlador photoCallTimer:** este controlador tiene su propio timer de 9 segundos y se encarga de cargar la escena photoCall cuando la animación de la cuenta atrás y el timer finalicen (acaban al mismo tiempo con una diferencia inapreciable).

- **Controlador pantalla *PhotoCall:*** este controlador se encarga de gestionar los siguientes eventos:
  - *Sacar fotografía:* para poder sacar la fotografía el controlador utiliza la librería JRPiCam que ofrece la posibilidad de ejecutar los comandos para controlar el módulo de la cámara. Una vez capturada la imagen y almacenada en un *BufferStream* se combina con el logotipo del evento para poder añadirlo como marca de agua en una esquina de la imagen y se carga la imagen combinada en el contenedor de la imagen en la interfaz. Para combinar las dos imágenes se ha cargado las dos imágenes en dos *BufferStream* diferentes y se ha utilizado un objeto *Graphics* para combinar las dos utilizando los parámetros de tamaño de cada imagen para re-dimensionar el logotipo y que se adapte a la imagen base.
  - *Botones:* hay dos botones de la interfaz el de volver a sacar otra fotografía y el de publicar en Facebook. El controlador se encarga de detectar si se pulsa alguno de estos botones. En el caso de volver a sacar otra fotografía se vuelve a la pantalla de la cuenta atrás y en el caso de publicar en Facebook se carga la escena UploadPhotoController.
- **Controlador UploadPhotoController:** este controlador también implementa la interfaz Obsever y tiene la función de publicar la fotografía obtenida en la escena photoCall y subirla a Facebook utilizando la función postPhoto. Para ello se utiliza el token que está asociado a la pulsera que se ha leído y a continuación se cargará la escena de la pantalla principal. En el caso de que el usuario no haya aceptado los permisos de publicación en su aplicación móvil se muestra una mensaje informándole de que debe aceptar tales permisos antes de poder publicar.

- **Vista** En esta sección se mostrará un conjunto de diagramas que detallan todos los elementos que forman cada uno de los interfaces de la aplicación.

  - **Pantalla principal:**

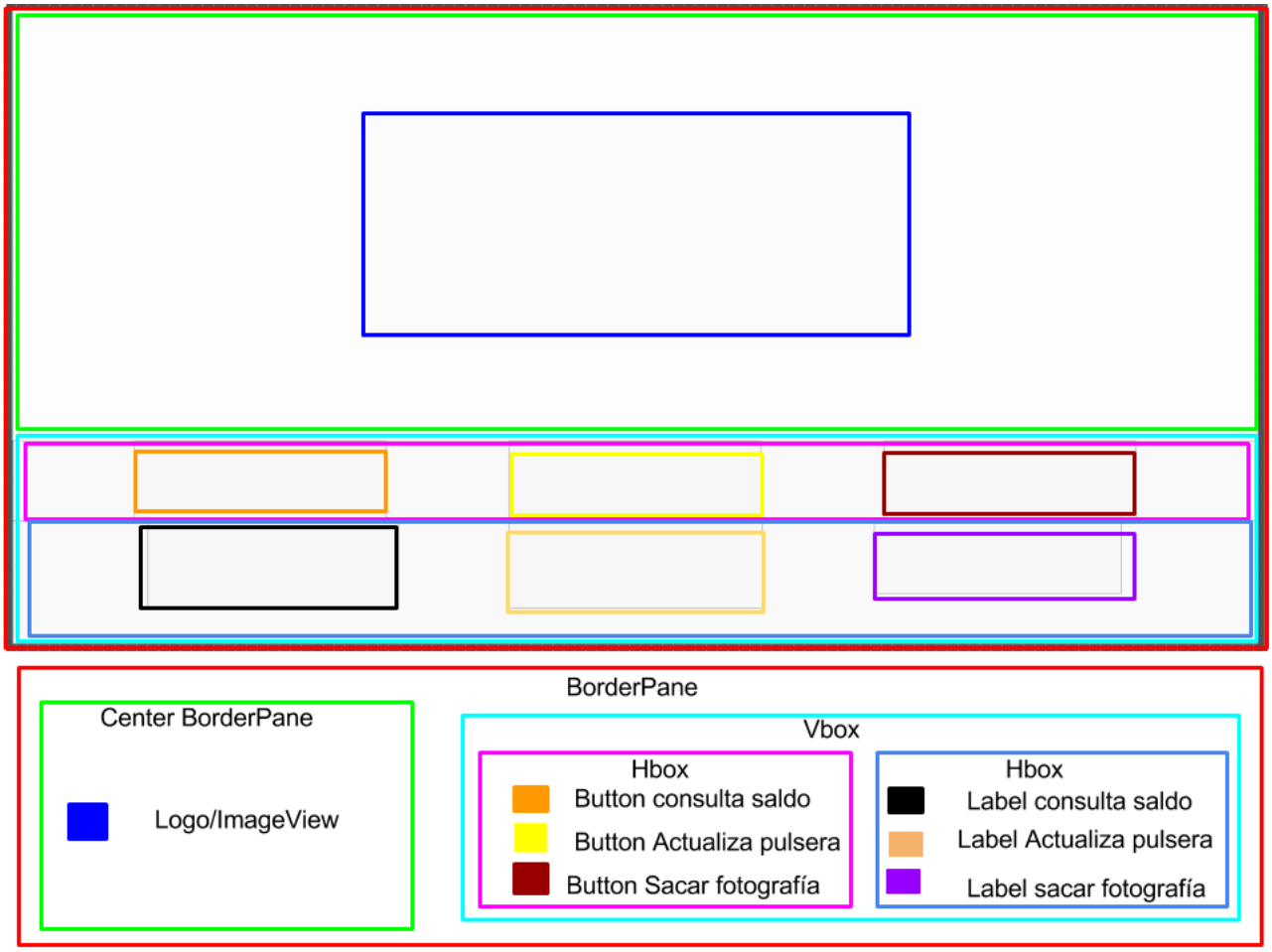


Figura 5.9: Estructura de la interfaz de la pantalla principal de la aplicación tótem

- **Pantalla de consulta saldo:**

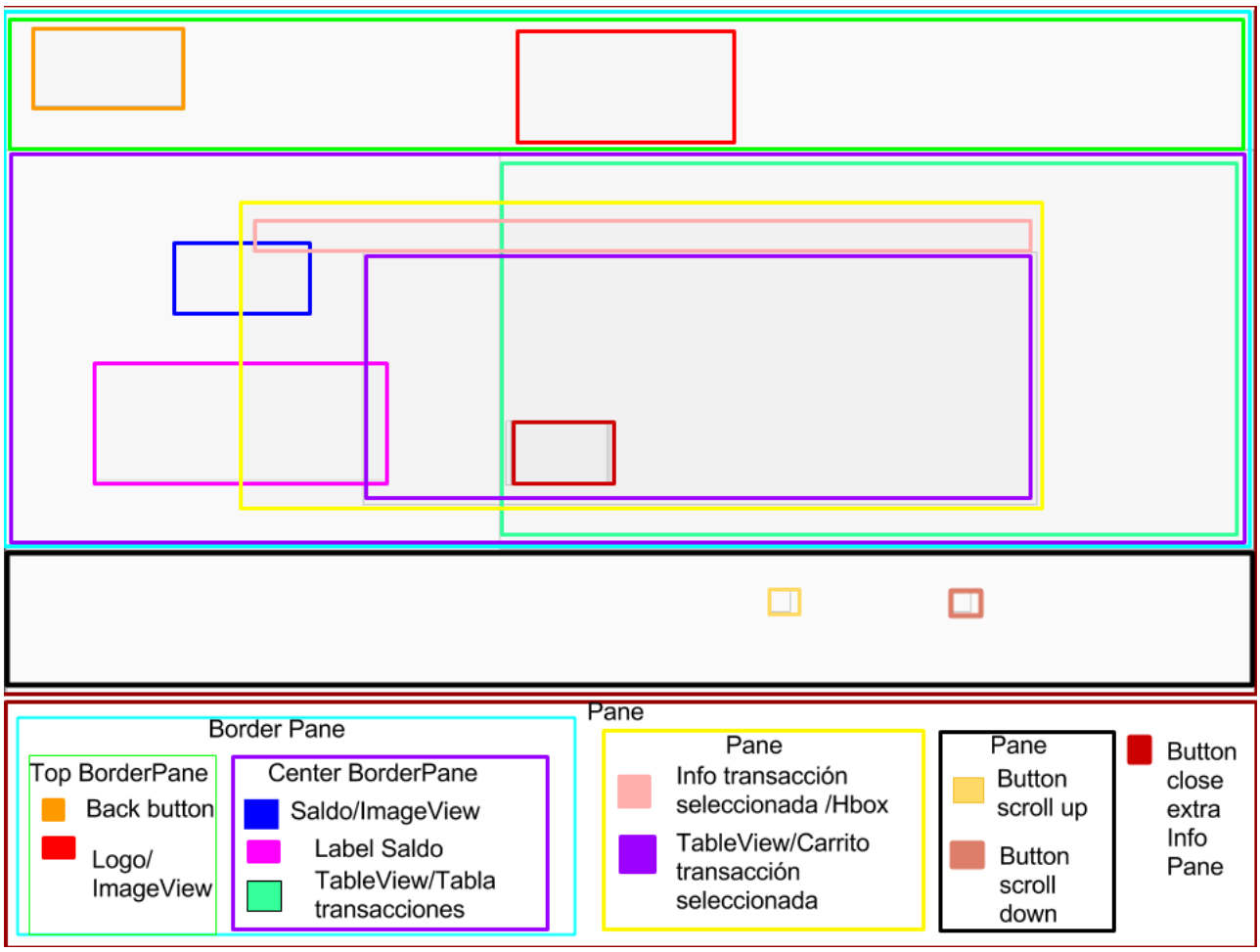


Figura 5.10: Estructura de la interfaz de la pantalla de consultar saldo de la aplicación tótem

- **Pantalla de lectura de la pulsera:**

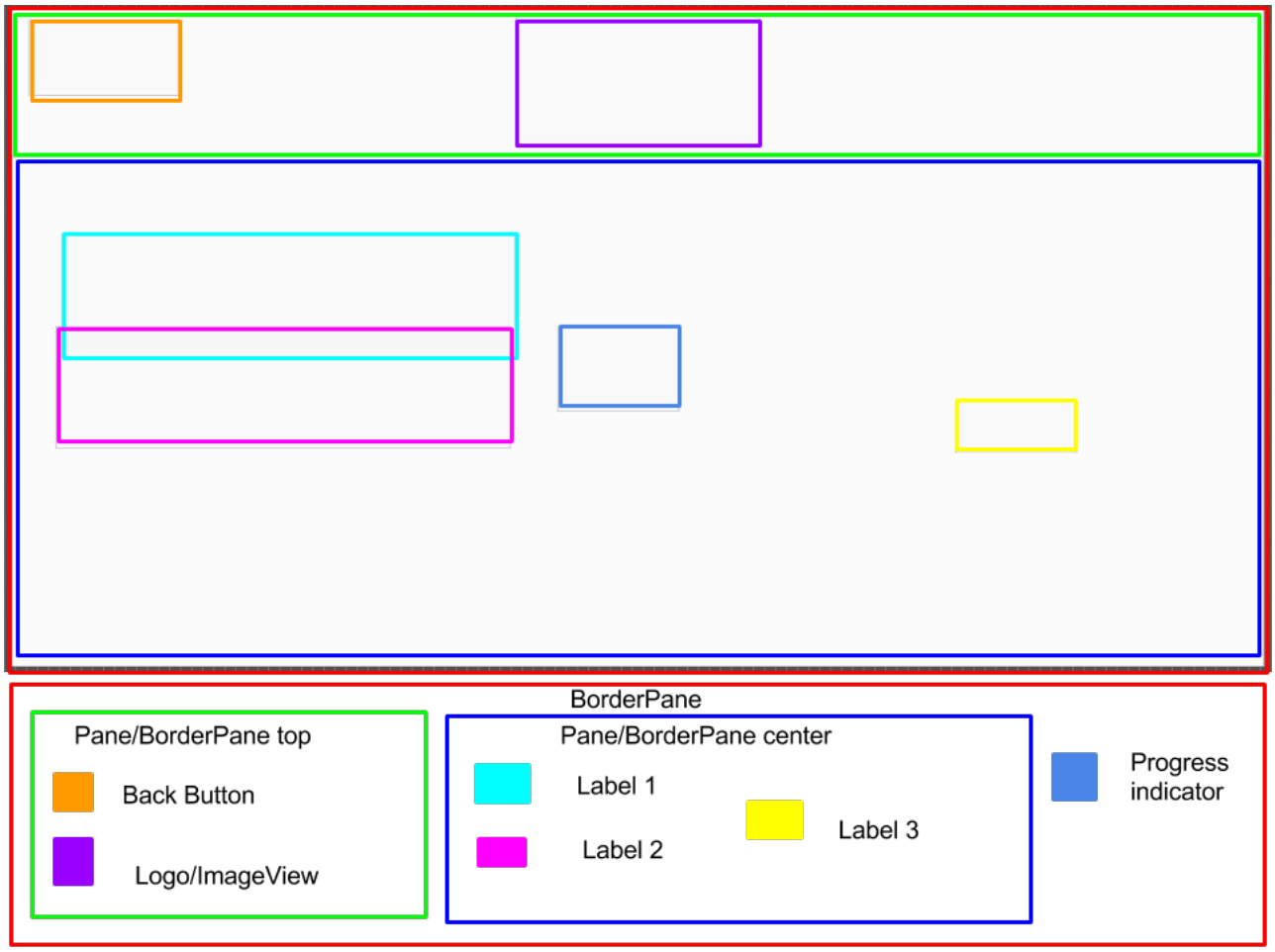


Figura 5.11: Estructura de la interfaz de la pantalla lectura de la pulsera de la aplicación tótem

- **Pantalla actualizar pulsera:**

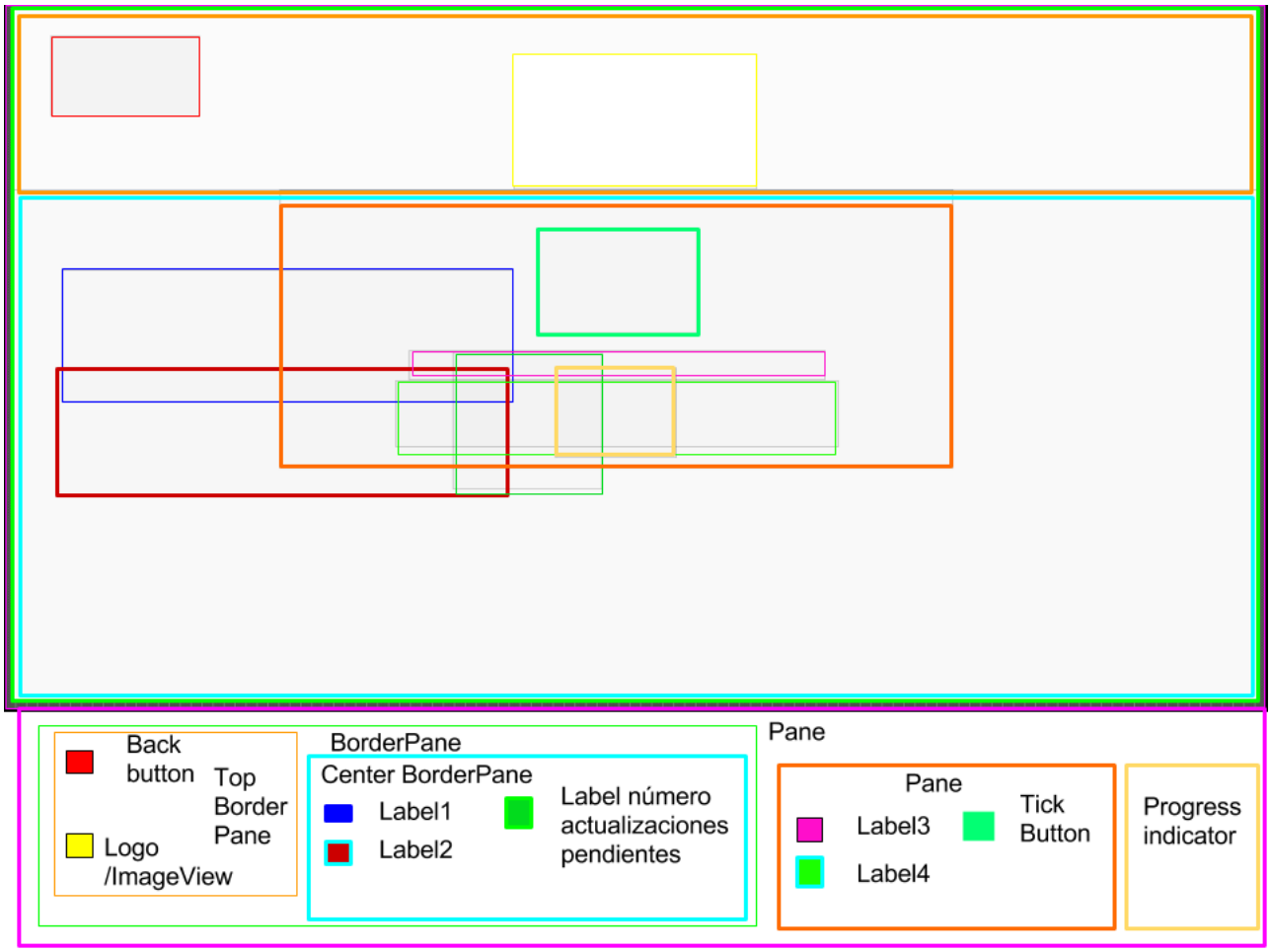


Figura 5.12: Estructura de la interfaz de la pantalla actualizar pulsera de la aplicación tótem

- **Pantalla cuenta atrás para sacar la fotografía:**

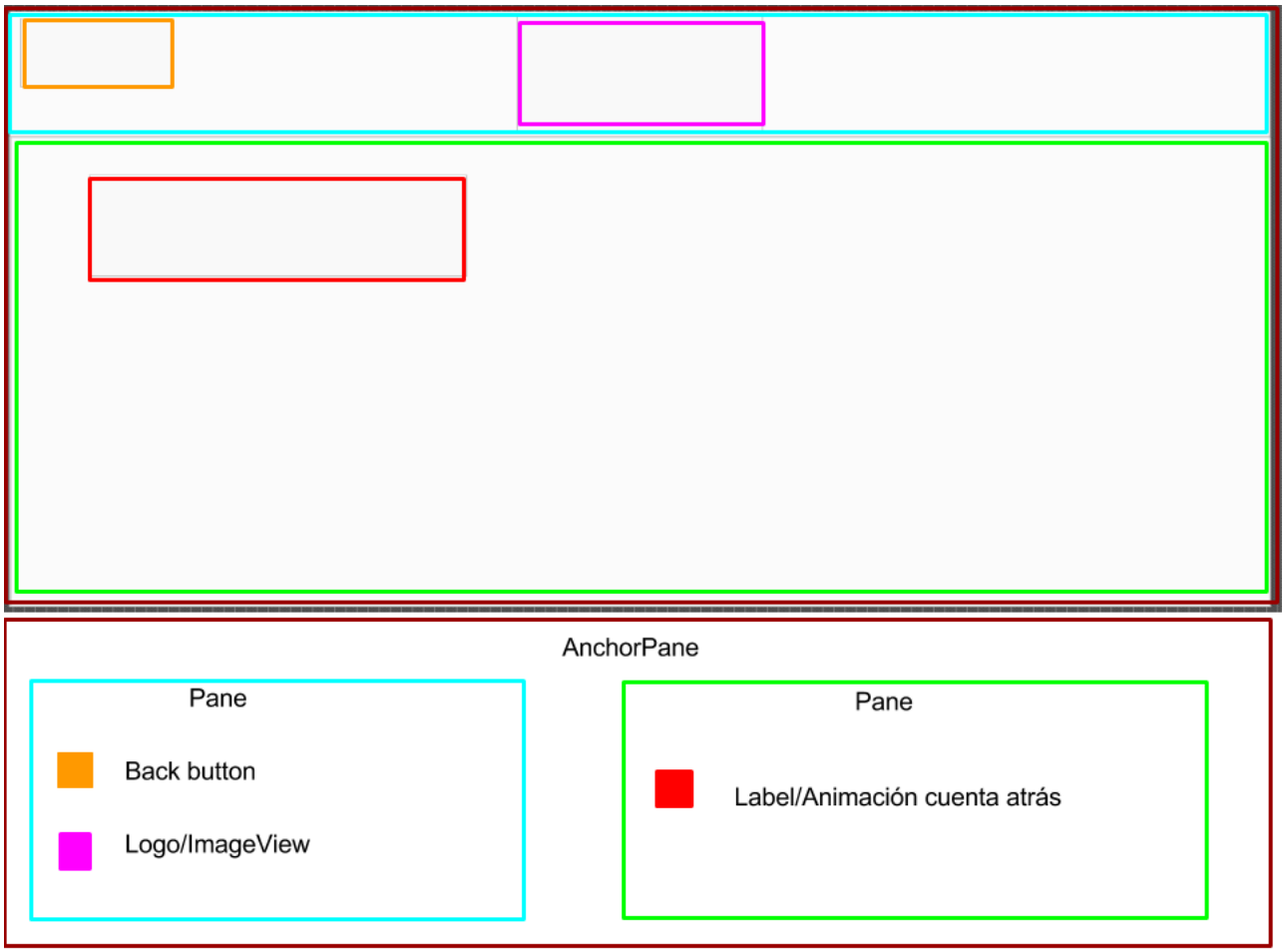


Figura 5.13: Estructura de la interfaz de la pantalla cuenta atrás de la aplicación tótem

- **Pantalla del *photoCall*:**

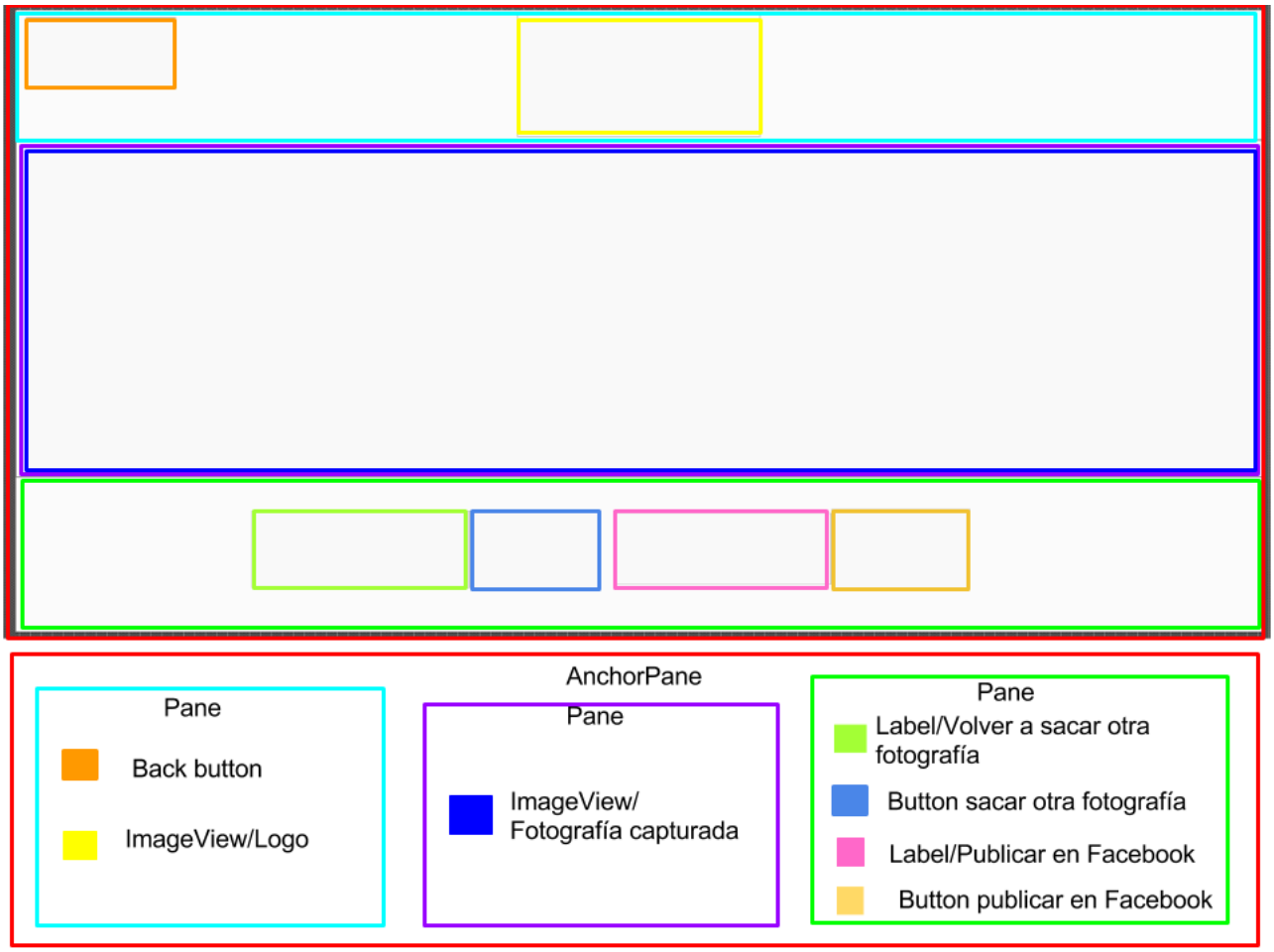


Figura 5.14: Estructura de la interfaz de la pantalla del *photocall* de la aplicación tótem

5.3.2. Verificación y validación

Como se ha comentado anteriormente en el documento la empresa utiliza la metodología TDD a la hora de desarrollar nuevo software. Por lo tanto, para toda la parte que forma el Modelo de la aplicación se han creado los test pertinentes antes de realizar la implementación y se han ejecutado para comprobar que todas las peticiones se envían y reciben correctamente.

Por ora parte, una vez ensamblados todos los componentes de *hardware* se ha realizado una prueba de rendimiento a lo largo del tiempo, donde se ha dejado un periodo considerable de tiempo la aplicación ejecutándose para observar su comportamiento y encontrar errores en la ejecución.

Finalmente la prueba definitiva se produjo durante el festival MadCool que se realizó durante los días 16-19 de junio donde se puso el producto final al servicio de los usuarios. Durante este evento se instalaron 18 tótem distribuidos por todo el recinto del evento que estuvieron en funcionamiento durante todo el evento. Los tótem fueron una atractiva opción para los usuarios y sobre todo destacaron por su popularidad las fotos realizadas en los tótem.

Gracias a esta oportunidad y después de haber escuchado y observado las opiniones , sugerencias e impresiones de los usuarios se ha llegado a la siguiente conclusión:

- **Simplificación de la aplicación:** durante el primer día pude observar las primeras reacciones y comportamientos cuando un usuario usaba el tótem por primera vez. Su primera acción era pasar la pulsera por el lector sin presionar ningún botón para seleccionar qué acción realizar (consultar saldo, actualizar la pulsera o sacarse una fotografía). Después de analizar cuál sería una posible mejora para este problema hemos decidido que en la siguiente versión de la aplicación la pantalla de consulta y actualización de la pulsera se van a fusionar en una sola. Lo que conllevará que lo único que tendrá que hacer el usuario será pasar la pulsera y le aparecerá toda la información de su saldo y sus transacciones y sólo si tiene actualizaciones pendientes se le mostrará en la misma pantalla un mensaje que le indicará que debe pasar su pulsera otra vez para ser actualizada. Por otro lado el proceso de sacarse la foto será el mismo dejando un botón para sacarse la foto en la pantalla principal.
- **Mejora del diseño físico del Tótem:** otra mejora planteada después del festival ha sido que se tendrá que rediseñar la estructura del tótem, ya que nos dio muchos problemas durante la instalación física de éstos, debido a que al fijarlos en el suelo los cables quedaban aplastados y tuvimos que montar unas plataformas improvisadas.

- **Problema con el lector NFC del Tótem:** durante el transcurso del evento detectamos un error de hardware en los lectores instalados en los tótem que ocasionaba que cada 50-60 lecturas aproximadamente el lector dejaba de leer y se tenia que ir reiniciar el tótem para que funcionara otra vez correctamente. Para solucionar este problema se realizó un script en bash que utiliza un programa en C (en la bibliografía está el link del repositorio Git del programa en C) para cortar momentáneamente la corriente de un puerto USB específico, simulando una desconexión física del lector. Utilizando este script y el planificador de tareas cron se planificó una tarea de desconexión del dispositivo cada 10 minutos, lo que solucionó provisionalmente el problema del lector.

  El script realizado es el siguiente: [7]

  ```
  #!/bin/bash
  sudo ./hub-ctrl -P 2 -p 0 # turn off port 4
  sudo ./hub-ctrl -P 2 -p 1 # turn on port 4
  ```

  Comando cron para planificar la tarea (se debe añadir esta linea al fichero de configuración crontab):

  ```
  */5  *  * * *   root    /root/resetNFCreader.sh
  ```

5.3.3. Configuración de la aplicación en la Raspberry pi 3

La aplicación se ejecuta con el SDK de Java 8. Para instalar Java 8 en Linux hay que seguir los siguientes pasos:

- **Add Java 8 PPA**:

  ```
  sudo vim /etc/apt/sources.list.d/java-8-debian.list ->
  deb http://ppa.launchpad.net/webupd8team/java/ubuntu trusty main
  deb-src http://ppa.launchpad.net/webupd8team/java/ubuntu trusty main
  sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys EEA14886
  ```

- **Install Java 8**

  ```
  sudo apt-get update
  sudo apt-get install oracle-java8-installer
  ```

- **Verify Java Version**

  ```
  java -version
  ```

- **Configure Java Environment**

  ```
  sudo apt-get install oracle-java8-set-default
  ```

Una vez instalado Java 8, para poder ejecutar la aplicación con todas sus funcionalidades, se utiliza el siguiente comando:

```
sudo optjdk1.8.0_77binjava
-Dcom.sun.javafx.touch=true
-Dcom.sun.javafx.isEmbedded=true
-Dcom.sun.javafx.vk.adjustwindow=true
-Dcom.sun.javafx.virtualKeyboard=javafx
-Djava.ext.dirs=optjdk1.8.0_77jrelibexthomepijrelibext
-Dsun.security.smartcardio.library=usrlibarm-linux-gnueabihflibpcsclite.so.1
-Dglass.platform=Monocle
-Dmonocle.input.31aad1110.minX=3933
-Dmonocle.input.31aad1110.minY=845
-Dmonocle.input.31aad1110.maxX=3155
-Dmonocle.input.31aad1110.maxY=267
-Dmonocle.input.31aad1110.flipXY=false
-jar homepiPhotoCall.jar 2&1  tee -a varlogtotem.log
```

Parámetros del comando de ejecución de la aplicación:

- **javafx touch:** esta variable habilita que se lea la entrada de la pantalla táctil.
- **javafx isEmbedded:** indica que se trata de un aplicación implementada en Javafx.
- **javafx vk adjustwindows:** activa la funcionalidad de que la interfaz de la aplicación se adapte a la pantalla.
- **javafx virtualkeyboar:** activa el teclado virtual proporcionado por Javafx
- **ext dirs:** indica que paquete de Java se va ha utilizar.
- **security smartcardio library:** indica qué librería se utiliza para gestionar la lectura de la pulsera.
- **Dqlass plataform:** se indica que se utilizará Monocle para gestionar la pantalla táctil.
- **Dmonocle minX,maxX,minY,maxY:** estos parámetros se utilizan para configurar la detección de la pantalla táctil y representa los valores recogidos anteriormente por la salida que produce la pantalla táctil al ser pulsada.

Capítulo 6

Conclusiones

El objetivo principal de este proyecto ha sido integrar las redes sociales en el proyecto Easygoband, lo cual se ha superado satisfactoriamente ofreciendo aplicaciones simples con una amplia funcionalidad y dejando abierto un abanico de ideas para proyectos futuros.

Por ejemplo, añadir a la aplicación del tótem la funcionalidad de poder retocar la fotografía antes de publicarla, añadirle un comentario y etiquetar a las personas que aparecen en la fotografía.

Cuando se planificó la plataforma en la que se ejecutarían las aplicaciones planteé utilizar un dispositivo llamado ODROID-C2 (nueva versión de la BeagleBone Black) ya que la Raspberry pi suele plantear problemas con la máquina virtual de Java y el ODROID-C2 es totalmente compatible con la distribución de Linux Ubuntu y por lo tanto no surgirían problemas inesperados. Pero al final se la empresa decidió utilizar la Raspberry pi 3. También se estuvo barajando la posibilidad de utilizar un web-cam para sacar las fotografías, pero planteé la opción del módulo cam de la Raspberry y fue éste al final el que utilizamos. Por otro lado la estructura del tótem necesita un rediseño físico para facilitar su montaje, mantenimiento y poder incorporar a la cámara un flash para poder sacar buenas fotografías con poca luz.

Por todo lo descrito en esta memoria, me ha parecido un proyecto interesante en el que he aprendido ha utilizar con más profundidad: las APIs de Facebook, Twitter e Instagram, el uso del GITLab, el uso de otro entorno de desarrollo similar a Eclipse, que utilizamos durante todo el grado, como es Intellij, Javafx y el uso y la configuración de la Raspberry pi.

Dejando de lado los aspectos técnicos aprendidos, he aprendido mucho del trabajo en equipo, la división de tareas y la metodología de programación ágil TDD. Mi estancia en la empresa ha sido muy agradable, ya que la gente ha sido muy amable y me han ayudado cuando he tenido algún problema y viceversa. Por último, la oportunidad de ver el proyecto en funcionamiento y poder interaccionar con los usuarios me ha ayudado mucho a mejorar y entender las necesidades de los usuario de forma más clara.

En conclusión estoy muy contento de mi estancia en practicas realizada en PaynoPain. Y estoy seguro que todo lo aprendido durante estos 3 meses de prácticas me servirá como base y experiencia para mis próximos proyectos personales.

Bibliografía

[1] Facebook, *Graph API de Facebook, visited on: 2016-07-20* , https://developers.facebook.com/docs/graph-api

[2] Facebook, *Gestion Token de larga duracación Facebok, visited on: 2016-07-20* , https://developers.facebook.com/docs/facebook-login/access-tokens/expiration-and-extension

[3] Instagram, *Documentación de la API de Instagram, visited on: 2016-07-20*, https://www.instagram.com/developer/

[4] Raspberry pi, *Documentación del modulo de la camara de Raspberry pi, visited on: 2016-07-20* , https://www.raspberrypi.org/documentation/usage/camera/README.md

[5] Andrew Dillon, *Modulo JRPiCam*, https://github.com/Hopding/JRPiCam

[6] Oracle, *Documentación desarrollador JavaFx, visited on: 2016-07-20*, https://docs.oracle.com/javafx/2/

[7] Joel Dare, *Programa en C que se encarga de controlar los puertos USB, visited on: 2016-07-20* , https://github.com/codazoda/hub-ctrl.c

[8] Twitter, *Documentación API Twitter, visited on: 2016-07-20* , https://dev.twitter.com/rest/public

Anexo A

Manual gráfico de usuario de la aplicación del Tótem

Manual de usuario

En este anexo se adjunta una manual gráfico de usuario que se ha realizado para poder mostrar de forma rápida y comprensible las funcionalidades que nos ofrece la aplicación del Tótem.

Este manual tiene tres secciones donde se muestra cómo consultar tu saldo, cómo actualizar tu pulsera y cómo sacarte una fotografía y compartirla en tu muro de Facebook.

A.1. Consultar tu saldo:

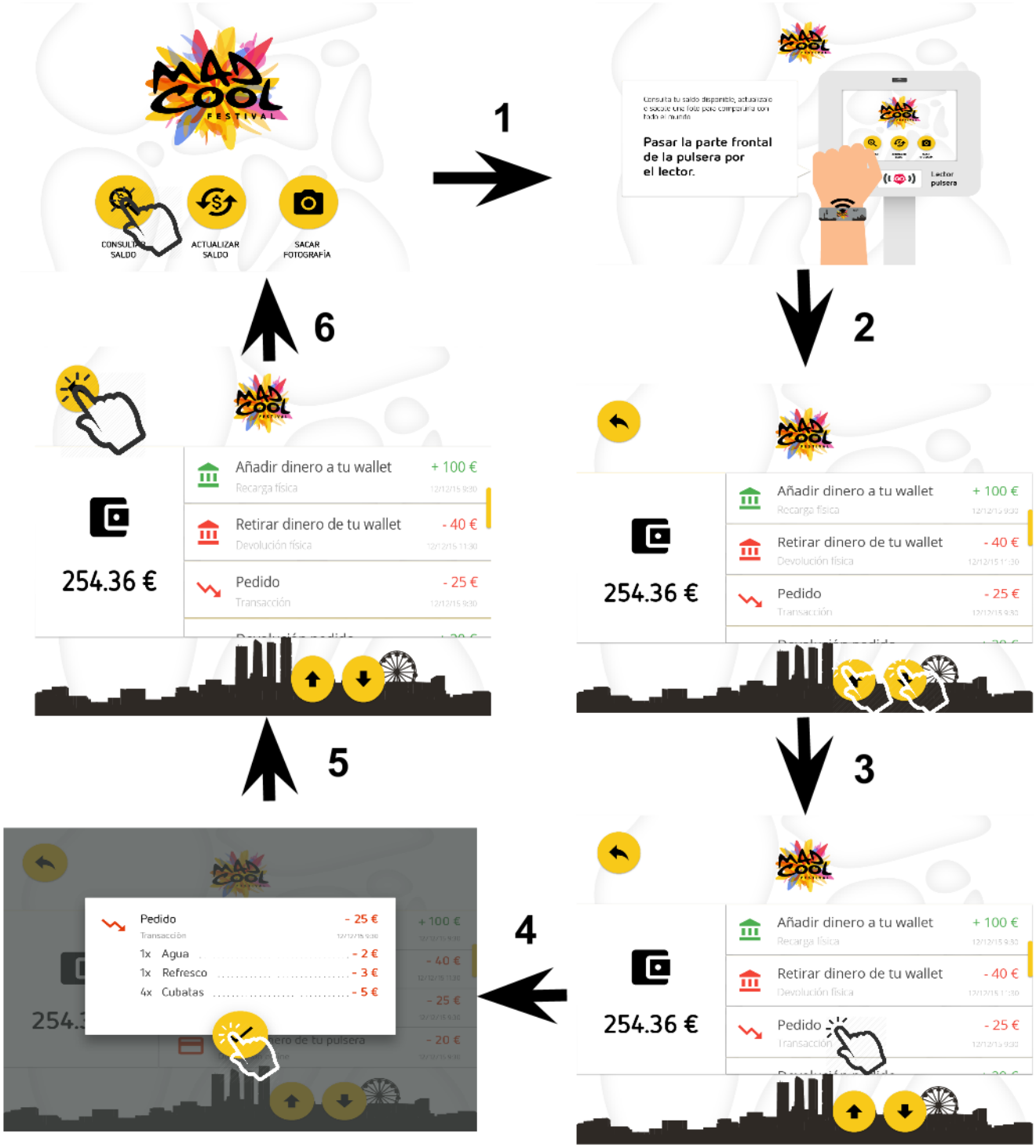


Figura A.1: Manual gráfico de usuario para consultar el saldo de tu pulsera

A.2. Actualizar tu pulsera:

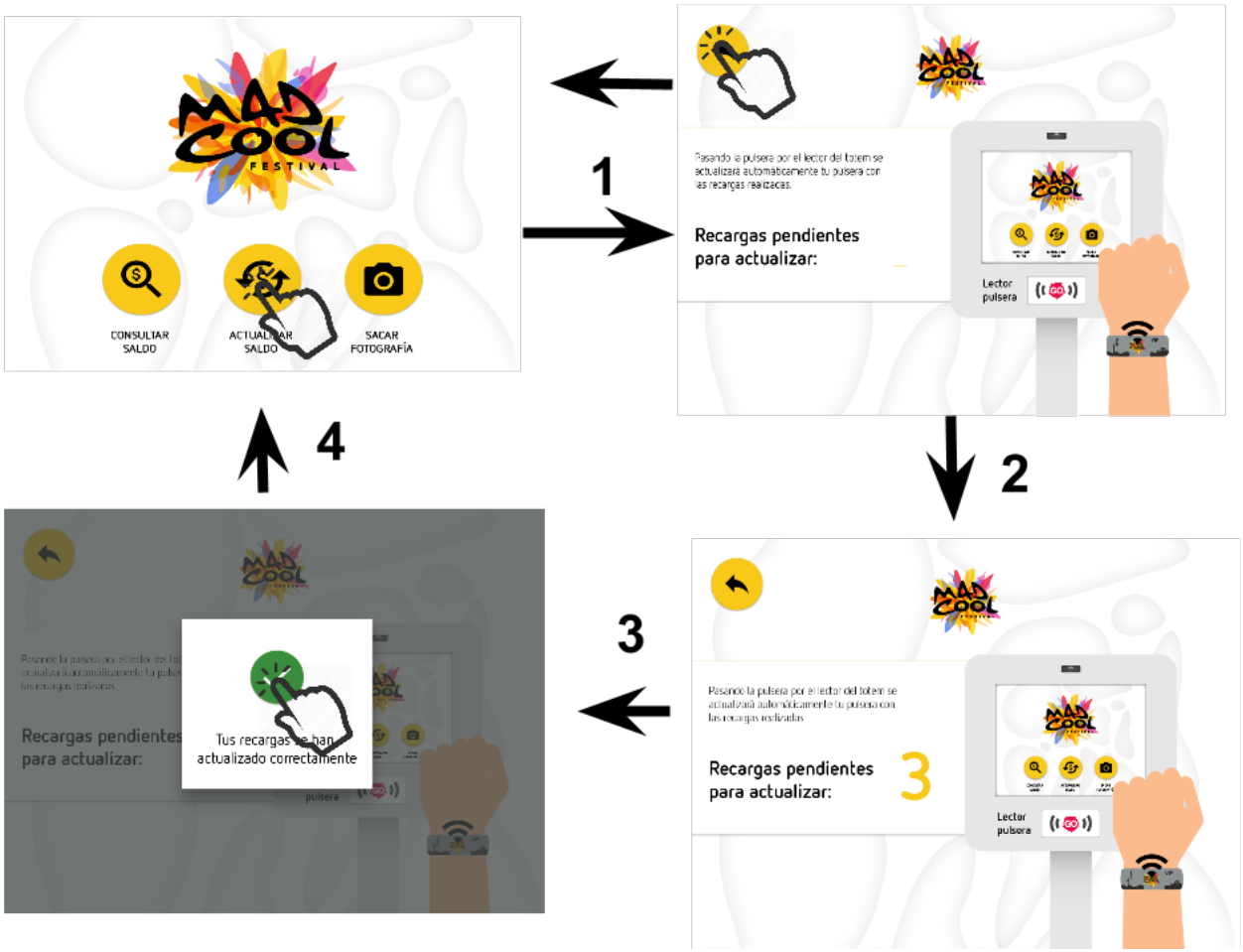


Figura A.2: Manual gráfico de usuario para actualizar tu pulsera

A.3. Sacarse una fotografía y compartirla en Facebook:

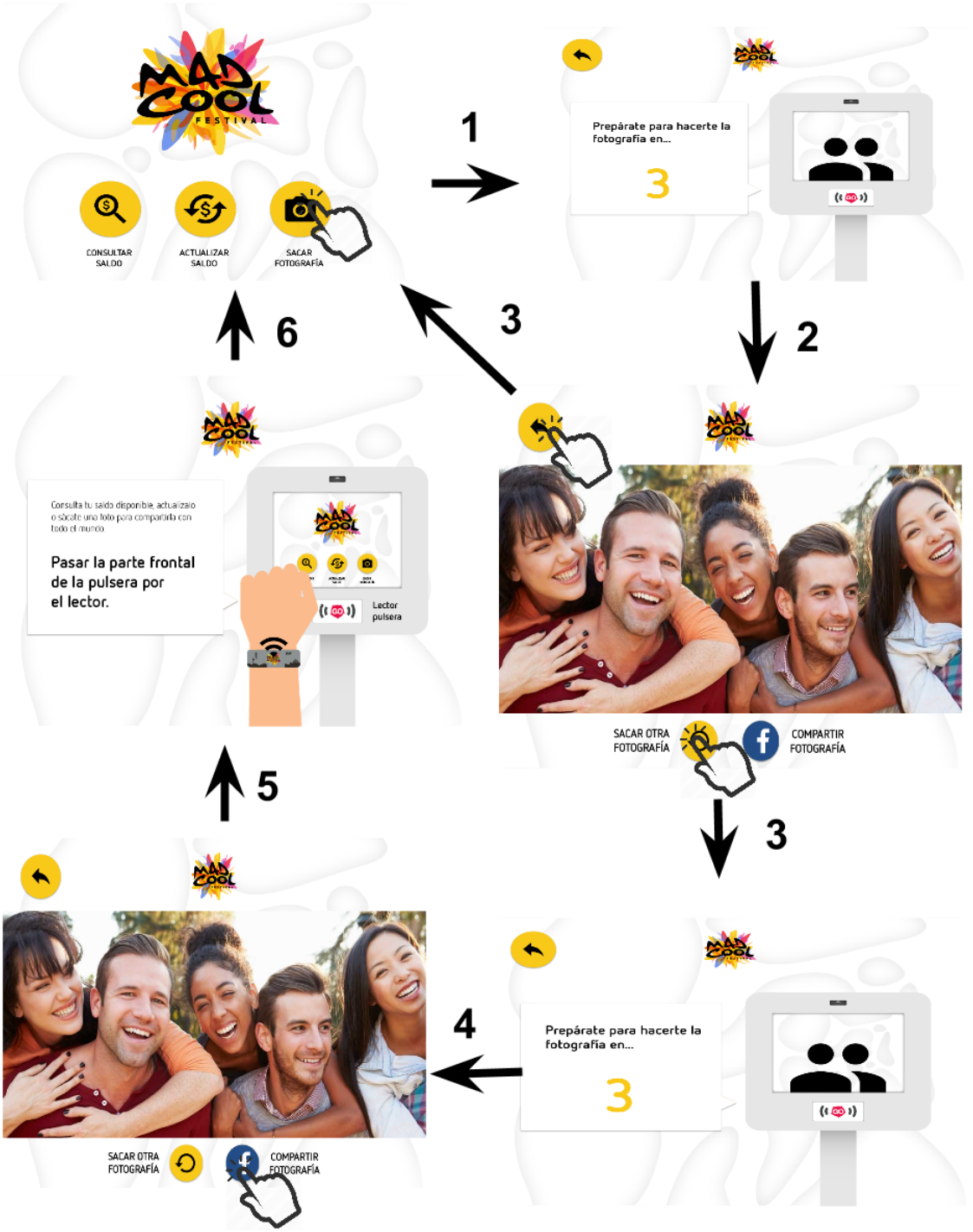


Figura A.3: Manual gráfico de usuario para sacarse una fotografía y compartirla en Facebook